**Final Performance Report**
**GRANT No. N00014-95-1-G037:**
**Modeling of Cosmic-Ray Transport Processes**

*Prepared by:*

*A. F. Barghouty, Principal Investigator*
*Physics Department*
*Roanoke College*
*Salem, VA 24153*
*E-mail barghouty@aleph.roanoke.edu*
*Tel. (540)375-2431; FAX (540)389-4236*

*October 1997*

*Submitted to:*

**Scientific Officer (3 copies):**
**Dr. Allan J. Tylka, Code 7654**
**Naval Research Laboratory**
**Washington, DC 20375-5326**



**Administrative Grants Officer (1 copy):**
**Office of Naval Research**
**Resident Representative (ONRRR)**
**101 Marietta Tower, Suite 2805**
**101 Marietta Street**
**Atlanta, GA 30323-0008**



**Director (1 copy):**
**Naval Research Laboratory**
**Code 2627**
**Washington, DC 20375-5326**

**Defense Technical Information Center (2 copies):**
**Building 5, Cameron Station**
**Alexandria, VA 22304-6145**

## Brief Background

Per the description of the Statement of Work of GRANT No. N00014-95-1-G037, this two-year grant was intended to fill a number of current and near-future needs of NRL's cosmic-ray basic research efforts as well as related applied research efforts like SEE (space-radiation environment studies) that have been developed or currently being extended by the cosmic rays group at NRL.

The support has been analytic and computational in nature in the general area of modeling of cosmic-ray transport in the heliosphere.

This final report describes the main support afforded by this grant to NRL's cosmic-ray group for the two years (9/95-9/97) of the grant. The main task, by far, has been the development, testing, and data comparison, of a global time-dependent and three-dimensional heliospheric transport code of galactic cosmic rays. While the code is based on current, standard and established theory of solar modulation of galactic cosmic rays, it is far more computationally demanding than what a typical application-oriented study may require (e.g., tasks with connection to SEE work). To such ends, the purpose of developing such a code was to afford the group a fully three-dimensional solar-modulation model by which a computationally efficient parametric set of simulated data (that can easily and efficiently be incorporated in larger semi-empirically based models like CREME96 (Tylka et al. 1997) for example) can reliably and efficiently be developed (e.g., Adams & Lee 1996).

This development should afford the group both qualitative and quantitative advantage in this regard and relative to modulation codes currently available to the group which tend to be rudimentary one-dimensional (so-called spherically-symmetric) codes.

The first section of this report describes in some detail the basic physical model the new three-dimensional transport code is based upon. The second section highlights the numerical implementation and various algorithms of the code, while the main routines are listed as an appendix. The third section illustrates some sample calculations and comparison to available data. The fourth section suggests some directions and recommendations for future work based on this code for applications-oriented studies by the group at NRL, as well as for more basic-physics oriented improvements. The fifth and final section is a list of cited references.

## 1. Solar Modulation of Galactic Cosmic Rays

### 1.1 The Physical Model

The standard model of long-term solar modulation of GCR that is used for our purposes in this work is the one based on Jokipii & Parker (1970) transport equation:

$$\frac{\partial U}{\partial t} = \nabla \cdot (\tilde{K}^s \nabla U) - \nabla \cdot \vec{V}_{sw} U - \langle \vec{V}_d \rangle \cdot \nabla U + \frac{1}{3}(\nabla \cdot \vec{V}_{sw}) \frac{\partial}{\partial T}(\alpha T U) \ . \tag{1}$$

where $U = U(\vec{r}, T, t)$ is the number density of GCR as a function of position $\vec{r}$, kinetic energy $T$, and time $t$. $U$ is related to the (solar-minimum) observable omni-directional cosmic-ray intensity $j$ as $j = vU/(4\pi)$, where $v$ is the particle's speed. The streaming flux vector $\vec{F}$ is then written

$$\vec{F} = -\tilde{K}^s \nabla U + \langle \vec{V}_d \rangle U + \vec{V}_{sw}[U - \frac{1}{3}\frac{\partial}{\partial T}(\alpha T U)] \ . \tag{2}$$

and the anisotropy vector is $\vec{\delta} = \vec{F}/vU$. Below we briefly describe the various terms in Eq. (1) and the associated physical processes they represent. $\alpha$ in Eqs. (1) and (2) is the standard scalar function $\alpha = (2m_o c^2 + T)/(m_o c^2 + T)$, with $m_o c^2$ being the rest-mass of the particle.

Eq. (1) is a dynamic-balance (i.e., time-dependent) statement for $U$ in three dimensions that is subject to the four fundamental physical processes that comprise the ***standard*** model of solar modulation: (1)*diffusion* (due to the irregular component of the heliospheric magnetic field); (2)*convection* (due to the outflowing solar-wind plasma carrying with it the frozen-in heliospheric magnetic field lines); (3)*drift* (due to the large-scale curvature and gradient of the regular component of the heliospheric magnetic field, and (4)*adiabatic energy loss* (due to the diverging solar-wind plasma).

The heliospheric magnetic field is taken to be composed of an irregular component superposed on a regular one, i.e., $\vec{B} = \vec{B}_o + \delta\vec{B}$, with $\nabla \cdot \vec{B} = 0$. We use the standard[1] description for the regular component of the heliospheric magnetic field, i.e., a multi-sector field with Parker's spirals and with opposing polarities above and below a wavy current sheet (Kóta & Jokipii 1983; and references therein):

$$\vec{B}_o(r, \theta, o) = \vec{B}_o(r, \theta)\ [1 - 2S(\theta - \theta')] \ . \tag{3}$$

where the single-polarity field $\vec{B}_o(r, \theta)$, note its $o$-independence, is written

$$\vec{B}_o(r, \theta) = A\ [\frac{1}{r^2}\ \hat{r} - \Omega_\odot \sin\theta / V_{sw} r\ \hat{\phi}] \ . \tag{4}$$

[1] As of yet, our numerical code does not take advantage of a recent suggestion (Jokipii & Kóta 1989; Smith & Bieber 1991) as well as observation (Smith et al. 1995) to modify the average field in the polar regions. This modification, however, can easily be made as well as others as outlined in §4.2 of this report.

$(r, \theta, \phi)$ refers to a heliocentric spherical coordinate system that we adopt here. $A$ is a constant that carries both the strength and polarity (depending on the solar cycle) of the single-polarity field, and $\Omega_\odot$ is the rotational speed of the Sun around its axis. $V_{sw}$ is the solar-wind speed with a radial but $\theta$-dependent velocity profile[2] :

$$\vec{V}_{sw}(\theta) = V_{sw}(\theta)\hat{r} = V_o(1 + \mu \sin^2 \lambda)\hat{r} \ , \tag{5}$$

where $\lambda$ is the solar geomagnetic latitude, $V_o$ is the solar-wind speed at zero latitude, and $\mu$ is a fitting parameter correlated with the solar cycle (Ananthakrishnan et al. 1995). Note that $\nabla \cdot \vec{V}_{sw} \neq 0$. $S(\theta - \theta')$ is a step function with $\theta'$ given by

$$\theta' = \frac{\pi}{2} + \sin^{-1}[\sin\alpha \ \sin(\phi - \phi_o + r\Omega_\odot / V_{sw})] \ , \tag{6}$$

and $\alpha$ is another fitting parameter related to the tilt angle of the current sheet at the Sun (also correlated with the solar cycle), and $\phi_o$ is an arbitrary constant. The current sheet is essentially defined by $S(\theta - \theta')$.[3]

The drift velocity vector averaged over near-isotropic pitch-angle scattering is written (Jokipii et al. 1976):

$$\langle \vec{V}_d \rangle = \frac{Pc\beta}{3q} \nabla \times (\vec{B}_o / B_o^2) \ . \tag{7}$$

where $P$ is the magnitude of the particle's momentum with charge $q$ and $\beta = v/c$, where $c$ is the speed of light. Note that $\nabla \cdot \langle \vec{V}_d \rangle = 0$, and $\langle \vec{V}_d \rangle$ is both polarity and charge sensitive via $qA$ being $> 0$ or $< 0$. Angle brackets in $\langle \vec{V}_d \rangle$ denote averaging of pitch-angle resonant scattering along the field lines due to the irregular component of the field.

The irregular component of the magnetic field is taken to be a random stationary field with zero mean, $\langle \delta \vec{B} \rangle = 0$, and characterized by a Kolmogorov-like power spectrum (i.e., $\propto \mathcal{F}\{\langle \delta\vec{B} \cdot \delta\vec{B} \rangle\} \propto k^{-\zeta}$) (Jokipii 1971), with $k$ being the wavenumber, $\zeta$ the Kolmogorov spectral index, and $\mathcal{F}\{\cdots\}$ denotes Fourier transform. The symmetric diffusion tensor in the local solar-wind frame is written:

$$\tilde{K}^s = \begin{pmatrix} \kappa_\perp & 0 & 0 \\ 0 & \kappa_\perp & 0 \\ 0 & 0 & \kappa_\parallel \end{pmatrix} \ . \tag{8}$$

[2]Unlike the discussion in Note 1, and although this dependence was not typically part of the standard modulation model, it was included in our numerical code early on due to the full time-dependent three-dimensional physical picture it was built to model. A second reason for its early inclusion had to do with our paying special attention to the question of inner boundary conditions and the so-called Péclet-spectrum analysis that we perform in our numerical solution of the convective-diffusive Eq. (1). §2.2

[3]See accompanying figure for a perspective of the magnetic current sheet in cartesian coordinates.

where, from quasi-linear theory,

$$\kappa_{\parallel} = \kappa_o \, R^{2-\zeta} \, \beta \left[ \frac{B_o^e}{B_o(\vec{r})} \right] . \tag{9}$$

with $R$ being the particle's rigidity, $B_o^e$ is the strength of the field at Earth, and $\kappa_o$ is a constant. The last term in Eq. (9) arises from requiring that the particle's gyroradius remains smaller than the scattering mean-free-path (Morfill & Völk 1979; Jokipii & Davila 1981) throughout the modulation region. We will have more to say about $\kappa_{\perp}$ in §1.2. In the heliocentric spherical coordinate system $\widetilde{K}^s$ becomes

$$\widetilde{K}^s = \begin{pmatrix} \kappa_{rr} & 0 & \kappa_{r\phi} \\ 0 & \kappa_{\theta\theta} & 0 \\ \kappa_{\phi r} & 0 & \kappa_{\phi\phi} \end{pmatrix} , \tag{10}$$

where, in terms of $\kappa_{\parallel}$ and $\kappa_{\perp}$,

$$\begin{aligned} \kappa_{rr} &= \kappa_{\parallel} \cos^2 \Psi + \kappa_{\perp} \sin^2 \Psi \, , \\ \kappa_{r\phi} &= (\kappa_{\perp} - \kappa_{\parallel}) \cos \Psi \sin \Psi \, , \\ \kappa_{\theta\theta} &= \kappa_{\perp} \, , \\ \kappa_{\phi r} &= \kappa_{r\phi} \, , \\ \kappa_{\phi\phi} &= \kappa_{\perp} \cos^2 \Psi + \kappa_{\parallel} \sin^2 \Psi \, , \end{aligned} \tag{11}$$

and $\Psi = \tan^{-1}(r\Omega_{\odot}/V_{sw})$.

To complete the physical picture of the standard modulation model one needs to specify physically meaningful initial and boundary conditions. We will, however, defer this discussion to §2.2, where, due to the full time-dependent and three-dimensional nature of the model and its numerical implementation, coupled with a rather stiff and complex transport PDE, we pay special attention to such conditions in our Péclet analysis section of the transport process.

### 1.2 Standard Quasi-Linear Cross-Field Diffusion

The theory of cross-field diffusion and its applications, e.g., to cosmic rays heliospheric transport, owes its inception to the works of Jokipii and Parker (Jokipii 1966; Jokipii and Parker 1969a; Jokipii 1973) where the mechanism was understood as mainly a non-resonant one due to the random-walk of the magnetic field lines themselves, so that test particles propagating along the field lines will also diffuse normal to the lines. Cross-field diffusion due to resonant scattering was shown to contribute little to the mechanism (Jokipii & Parker 1969a). [Note that the same mechanism was also being studied in the context of plasma physics and applications to Tokamak-field turbulence (Rosenbluth et al. 1966; for a recent review see, e.g., Isichensko 1992).]

In addition to application to cosmic rays heliospheric transport, which is the focus of this work, cross-field diffusion also plays significant roles in other astrophysical applications like diffusive shock-acceleration theories (Drury 1983; Lagage & Cesarsky 1983; Blandford & Eichler 1987; Jokipii 1987; Jokipii & Morfill 1987; Achterberg & Ball 1994; Duffy et al. 1995; Giacalone & Jokipii 1996) and cosmic rays galactic transport (Jokipii & Parker 1969b; Skilling et al. 1974; Ptuskin 1979; Barge et al. 1984; Corbelli & Veltri 1989; Chuvilgin & Ptuskin 1993; Achterberg & Ball 1994; Giacalone & Jokipii 1994; Dendy et al. 1995; Duffy et al. 1995; Klepach 1995; Ptuskin 1995).

The two essential ingredients in the theory of cross-field diffusion are the separation rate of the two (initially close) wandering field lines and the scaling of the perpendicular diffusion coefficient with the strength of the magnetic turbulence. The separation rate is understood to be a critical length scale in the problem as the two lines (and test particles that are tied to them) tend to become independent beyond this length scale. Early estimates of the separation rate (Jokipii 1973; Ptuskin 1979; Barge et al. 1984) give a rate on the order of the inverse of the turbulence correlation length. [More recent numerical (Zimbardo et al. 1995; Zimbardo & Veltri 1995) and analytic (Barghouty & Jokipii 1996) estimates suggest a more complex separation rate that, in addition to the correlation length, is coupled to the strength of the turbulence in a non-trivial, multi-region fashion. See §4.2 for more on the role of nonlinear cross-field diffusion, as opposed to quasi-linear description, in GCR modulation modeling.]

The standard quasi-linear description (Jokipii 1971) for the random walk of the field lines, in the limit when the scattering mean free path is $\gtrsim$ the turbulence correlation length $\ell$, is expressed as

$$\frac{\langle \Delta x^2 \rangle}{\Delta z} = \frac{1}{B_o^2} \int_{-\infty}^{\infty} dz' \; c_{xx}(0,0,z') = \frac{\mathcal{P}_{xx}(k=0)}{B_o^2} \; , \tag{12}$$

with $c_{xx}$ being the correlation function of $\delta B_x$ taken in the limit $\Delta z \gg \ell$, and $\mathcal{P}_{xx}(k=0)$ is the corresponding power spectrum at zero wavenumber. $\kappa_\perp$ is then given by

$$\kappa_\perp \sim \frac{v}{B_o^2} \mathcal{P}_{xx}(k=0) \; . \tag{13}$$

[The quasi-linear scaling is evident here in that the diffusion coefficient scales with $\delta b^2$.] Comparing the above expression to a similarly derived expression for $\kappa_\parallel$ at high rigidities

$$\kappa_\parallel \sim \frac{R^{2-\zeta_\beta}}{\mathcal{P}_{xx}(k=0)} \; , \tag{14}$$

and using an extrapolated estimate for $\mathcal{P}_{xx}(k=0)$, one is led to qualitatively conclude that

$$\kappa_\perp \ll \kappa_\parallel \; . \tag{15}$$

except, perhaps, for very small rigidities where the above expressions do not apply.

Thus in standard modulation models of GCR it has become a standard practice to assume that the magnitude of $\kappa_\perp$ is simply only a small fraction of $\kappa_\parallel$, typically $\lesssim 10\%$, i.e., and apart from the actual numerical ratio one would use for $\kappa_\perp/\kappa_\parallel$, $\kappa_\perp$ is essentially coupled to $\kappa_\parallel$. Therefore, and while we do not address more recent suggestions for variants of this assumption (e.g., anisotropy in $\kappa_\perp$, modifications due to larger fields at high latitudes, etc. (see discussions in, e.g., Kóta & Jokipii 1995, Bieber et al. 1995), for our purposes here, it must be kept in mind that the quasi-linear scaling is what is typically assumed in standard modulation models, as in the above standard quasi-linear expressions, as well as in variations thereof, independent of the assumed numerical ratio $\kappa_\perp/\kappa_\parallel$.

In addition to the standard quasi-linear scaling alluded to above, the standard separation rate as deduced from quasi-linear theory (Jokipii 1973; Barghouty & Jokipii 1996) is dependent both on the so-called large-step diffusion coefficient, $D_\infty \equiv \langle \Delta x^2 \rangle / \Delta z$ as given in Eq. (12) above, with its quasi-linear scaling embedded in, as well as on the parallel length scale in the turbulence $\ell$. To illustrate this for a turbulent field with truncated Kolmogorov-like power spectrum, i.e.,

$$P_{xx}(\vec{k}) = \int d^3r \exp(i\vec{k}\cdot\vec{r})\; c_{xx}(\vec{r}) \propto \frac{(k_y^2 + k_z^2)}{(k^2 + \ell^{-2})^{\zeta/2+2}}\; \exp(-k^2\ell_i^2)\;, \tag{16}$$

with $\vec{k}$ being the wavevector and $\ell_i$ ($\ll \ell$) being the turbulence inner scale, the Fokker-Planck diffusion coefficient $D_{FP}(\rho) \equiv \langle \Delta\rho^2 \rangle / \Delta z$, where $\rho \equiv |\delta\vec{r}_\perp| \ll \ell$ and $\delta\vec{r}_\perp = (\delta x, \delta y)$ is the 2-d separation vector between the two lines in the $x$-$y$ plane, can —to first-order in $\delta\ell^2 \equiv (\ell_i/\ell)^2$— be written as (Barghouty & Jokipii 1996)

$$D_{FP}(\rho) \approx 2D_\infty \left\{ 1 - \frac{2}{\Gamma(\zeta/2)} \left(\frac{\rho}{2\ell}\right)^{\zeta/2} \left[ \epsilon_1 K_{\zeta/2}\left(\frac{\rho}{\ell}\right) - \epsilon_2 K_{\zeta/2-2}\left(\frac{\rho}{\ell}\right) \right] \right\}\;, \tag{17}$$

where $K_\nu(Z)$ is the Bessel function of an imaginary argument and $\Gamma$ is Euler's gamma function. $\epsilon_1$ and $\epsilon_2$ are expansion coefficients in $\delta\ell^2$ given by

$$\epsilon_1 = 1 + \frac{\delta\ell^2}{(2 - \zeta + 4\delta\ell^2)}\;,\quad \epsilon_2 = \frac{\delta\ell^2(3+\zeta)}{(2-\zeta+4\delta\ell^2)}\;. \tag{18}$$

In short, by *standard* parameterization of the diffusion coefficient and the separation rate in the quasi-linear description of the random walk of the field lines we are referring to a scaling $\propto \delta b^2$ and a separation rate that depends on a parallel correlation length of $\delta\vec{B}$.[4] Further, the statistics is inherently Gaussian and the geometry is slab-like. Taken together, the quasi-linear description

[4] For a magnetic field in $x, y, z$ cartesian coordinates with a uniform part $B_o$ along $z$ and a normal fluctuating one $\delta\vec{B}(x,y,z)$, $\vec{B} = B_o\hat{z} + \delta\vec{B}$, with $\langle\delta\vec{B}\rangle = 0$ and subject to $\nabla\cdot\vec{B} = 0$, the relative turbulence strength is defined as $\delta b \equiv (\langle\delta\vec{B}\cdot\delta\vec{B}\rangle)^{1/2}/B_o$, where angle brackets denote ensemble averaging.

of cross-field diffusion is, therefore, inherently a classical brownian-motion description of the field lines.

## 2. Numerical Implementation

### 2.1 Numerical Solution of the Transport Equation in 1+4 Variables

We have developed our own numerical solution (Barghouty 1997) that integrates over $t$ Eq. (1) in three spatial dimensions $(r, \theta, \phi)$ and the kinetic-energy variable $T$, i.e., a full numerical solution in 1+4 variables. In this section we briefly describe the method by which our numerical solution is arrived at since it deviates from some of the earlier developed ones for this problem (we are referring here to solutions based, e.g., on Crank-Nickolson techniques and alternating direction implicit (or ADI) schemes with either momentum or energy as pseudo-time (e.g., Fisk 1971, 1976; Perko & Fisk 1983).)

Our numerical code integrates Eq. (1) using the so-called numerical method of lines (Scheisser 1991) in which the time variable is kept "continuous" and the finite-differencing scheme is applied to the other four variables at different points along a "time-line" which is then integrated. The advantage from separating the time variable from the rest is that the stability issue of the solution is, from the outset, separated from the accuracy issue. For a PDE like Eq. (1), simple eigenvalue tests can reveal the stiffness of the equation (this stiffness can also be readily appreciated by the many orders-of-magnitude separation in the strengths of the various diffusion-tensor terms at widely separated radial points), so that the stability issue is of concern.

All first-order and second-order diffusion-related derivatives of the dependent variable $U$ on the RHS of Eq. (1) are evaluated at a certain $t$ using a five-point centered differencing scheme (i.e., accurate to fourth-order in the step size of the respective variable). All first-order convection-related derivatives are evaluated using a five-point either upwind or downwind biased (depending on the variable) differencing scheme. All field, diffusion tensor, and drift vector terms along with their first and second order derivatives are evaluated analytically. Once all derivatives of $U$ all collected as the RHS of Eq. (1) at all $t$ points along the time-line, they are integrated using the (truly unconditionally stable and extremely efficient) sparse Jacobian-matrix technique of Hindmarsh (1982) (see also Byrne & Hindmarsh 1987) appropriate for stiff PDEs. The accuracy in this integration scheme is completely under the control of the user.

### 2.2 Boundary Conditions and Péclet Analysis

"Initial condition" for the purpose of reaching a stationary solution means at $t = 0$ we assume that $U$ takes on the unmodulated LISM isotropic distribution at the assumed outer heliospheric boundary of $r_\infty = 100$ AU, of the the form $U(r_\infty, \theta, \phi, T) = U_\phi(T) = \text{const}\ (m_o c^2 + T)^{-\eta}$ where $\eta$ is the typical LISM-spectrum exponent 2.65-2.75. This "initial condition" assumption also serves as the boundary condition for $U$ at $r_\infty = 100$ AU. The inner heliospheric boundary condition at $r_o =$

.01 AU is either taken to be $U(r_o, \theta, \phi, T) = 0$, or the number-density conserving condition according to Liouville's theorem, i.e., $\vec{F}(r_o, \theta, \phi, T) = 0$ (which was assumed in the sample calculations of §3) depending on the level of analysis. For Eq. (1) this condition is expressed as

$$\frac{\partial U}{\partial t}\Big|_{r=r_o} = -\frac{1}{3}\vec{V}_{sw} \cdot \nabla\Big[\frac{\partial}{\partial T}(\alpha T U)\Big]\Big|_{r=r_o} . \tag{19}$$

Both conditions seem to give very similar results at 1 AU, but this is not assured to be the case throughout the modulation region nor for all energies. For boundary conditions in the angle variables, $\partial U/\partial\theta = 0 = \partial U/\partial\phi$ are assumed periodic over the range $[0, \pi]$ for $\theta$ and $[0, 2\pi]$ for $\phi$. Because of the complete symmetry of the solution about the current sheet, the range for $\theta$ is taken to be $[0, \pi/2]$ thereby treating the inner singularity in the solution (when $\vec{B} \to 0$ at all points on the sheet) as a boundary condition for $\theta$. The singularity at $\theta = 0^o$ is treated using l'Hospital rule. For the "boundary" condition in $T$ we simply assume that $U(r, \theta, \phi, T_\infty) = U_0(r_\infty, \theta, o, T_\infty)$, with $T_\infty = 100$ GeV/Nucleon.

We perform a Péclet analysis of Eq. (1), and as we briefly describe below, the analysis touches upon both the viability of the numerical solution, on the one hand, as well as the assumed boundary conditions, on the other. Péclet number (or spectrum for energy-dependent transport coefficients as in Eq. (1)) is defined as the ratio of the convection group to the diffusion group in the PDE times the characteristic length scale in the problem. For Eq. (1), for example, we can define the Péclet spectrum for the $\theta$ and $r$ convection and diffusion groups as

$$\mathcal{P}_\theta(\theta, T) = \big|\langle\vec{V}_d\rangle_\theta(r^*, \theta, \phi^*, T)\big| r_\infty/\kappa_{\theta\theta}(r^*, \theta, \phi^*, T) , \tag{20}$$

$$\mathcal{P}_r(\theta, T) = \big|\langle\vec{V}_d\rangle_r(r^*, \theta, \phi^*, T) + V_{sw}(\theta)\big| r_\infty/\kappa_{rr}(r^*, \theta, \phi^*, T) . \tag{21}$$

for some $r^*$ and $\phi^*$. The significance of this number is that when it is too high ($\gg 1$), indicative of a strongly convective PDE, one needs to pay special attention to the behavior of the numerical solution at both exterior and interior boundaries as it can admit spurious discontinuities which can propagate throughout the characteristic length. In other words, the PDE can (numerically) be made to resemble more an advection equation with its usual reflective properties, but with spurious implications for its diffusive properties.

In numerical studies of Eq. (1), and especially with issues related to diffusion in three dimensions, this, in turn, may lead to erroneous and numerically-induced spurious "physical" features in the transport (e.g., Schiesser 1996). To illustrate, in Fig. 1 we plot $\mathcal{P}_\theta(\theta, T)$ for H calculated at 1 AU but averaged over $\varphi$. What Fig. 1 seems to indicate is that except for high-energy protons and close to the sheet in the ecliptic, the $\theta$-transport is weakly convective (or mostly diffusive). Contrast this with the $\mathcal{P}_r(\theta, T)$ spectrum in Fig. 2 for the radial groups where for low-energy protons and essentially for all $\theta$ the $r$-transport is primarily convective.

## 3. Sample Calculations and Comparison to Data

To give a feel for the high efficiency of the integration algorithm we use, for the sample calculations we show below and using a grid size of $15 \times 15 \times 15 \times 15$ (radial and energy grid points are logarithmically spaced), i.e., 50,625 ODEs to be solved in general, and with a prescribed relative accuracy of 1% (for time-integration), the method required the actual evaluation of only 11,998 ODEs (or grid-point visits) due to the sparsity of the Jacobian matrix of the ODE system (about 24% of total number of ODEs that other methods, generally speaking, may have to solve). With increasing grid size, the efficiency gets even better as the sparsity increases and an order of magnitude increase in efficiency is not untypical. Reaching a stationary solution required a mere 67 points along the time-line for a total CPU-elapsed time of 3.1 hrs (or about 23 ODEs per sec) on a DEC-ALPHA 250 machine.

For illustration purposes (our focus is on solar-minimum steady-state solutions) Eq. (1) is integrated until a stationary solution is reached ($t \lesssim 0.15$ yr) with the prescribed accuracy. Time-sensitive solutions are, naturally, also available. On a relatively coarse grid, Fig. 3 shows a sample calculated $\phi$-averaged H flux at 1 AU and $\theta = 90^o$ (in units of particles/m$^2$-sr-s-MeV/Nucleon) at solar minimum. Data from the 1977 and 1987 solar minima are also shown (even though the calculations pertain to $qA > 0$ ,i.e., 1977 solar-minimum). Table 1 lists the salient assumed physical parameters in the illustrations. Fig. 4 shows the calculated solar-minimum H flux as a function of the polar angle $\theta$ at 1 AU for three different energies where relaxation to isotropy with increasing energy is evident. In Figs. 5 and 6 the local $\theta$ and $r$ gradients are plotted for H at solar minimum.

## 4. Recommendations for Future Work

### 4.1 Semi-Empirical Modeling

Semi-empirical expressions for an effective long-term modulation of galactic cosmic rays can be put forward under the assumptions of steady-state and spherical symmetry conditions as has been done by, e.g., Evenson et al. (1983), Garcia-Munoz et al. (1985), and more recently by Adams & Lee (1996).[5] In these semi-empirical expressions convection, diffusion and adiabatic energy losses [but no drift effects][6] are all lumped into an effective diffusion coefficient $D(r, T)$, so that one can

[5] One can even use an effective modulating potential (Gleeson & Axford 1968) in a numerical simulation as has been done by, e.g., Letaw et. al. (1984).

[6] It is possible to heuristically include such effects in this semi-empirical context by lumping the radial component of the drift velocity vector to the radial solar-wind velocity vector. However, this drift addition is due to all spatial inhomogeneities in the regular heliospheric magnetic field and not just along the radial direction. As such, and in a spherically symmetric picture of the modulation region one is not justified in including only the radial component of the drift while ignoring the others!

arrive by direct integration at a simple expression of the form

$$U(r,T) = U_o(T) \exp\left[\int_{r_o}^{r} dr V_{sw} D^{-1}(r,T)\right] . \qquad (22)$$

where the spatial dependence of $D(r,T)$ is separated from its $T$ (or rigidity) dependence and is made to account for the adiabatic energy losses.

Semi-empirical expressions like the one above appear to be insensitive to the assumed outer boundary of the modulation region as well as to the exact form by which $D(r,T)$ depends on both $r$ and $T$. Also, such expressions appear to give reasonable fit to near Earth data over the 11-yr solar-cycle, i.e., not just for solar-minimum conditions, when $D$ is allowed to take on different values over the cycle, i.e., taking $D$ as function of time in addition to $r$ and $T$. Modulation studies using such semi-empirical expressions tend to converge to the main conclusion that, in sofar as near Earth data are concerned, the steady-state, spherically-symmetric modulation model of galactic cosmic rays is not an unreasonable one.

[For near Earth data one needs to keep in mind that this regime of the overall modulation region represents but a deep and hence asymptotic (in the mathematical sense of the transport equation and in the sense of the distance that the galactic cosmic-ray particle has to travel from the outer boundary of the heliosphere to near Earth) regime of the solution both in space and time. Thus, it may not be all that surprizing that the steady-state assumption appears to be a reasonable one. The spherical-symmetry assumption, on the other hand, is less obvious. But one can still argue that at lower energies ($\lesssim$ GeV/Nucleon), where the modulation is much more pronounced and significant, adiabatic energy losses dominate over the diffusive (and other) aspects of the transport process and as a result the modulation is less sensitive to variations in the diffusion terms (as well as assumed geometry), especially so for near Earth observations.]

One possible semi-empirical direction one can take is to use the simulated results of the 3D code to build a data base of the simulated modulating scalar factor $U(\vec{r},T)/U_o(T)$ throughout the heliosphere and as a function of rigidity. This way any transported particle at any given point $\vec{r}$ is assigned a factor depending on its rigidity. Computationally this should be quite efficient, if not all that robust. Semi-empirically, one can think of this modulating scalar function as similar to the one appearing in Eq. (22) in the lowest order, i.e., in the sense of expansion in $(r,\theta,\phi)$.

### 4.2 Improvements to the Physical Model

#### 4.2.1 Nonlinear Cross-Field Diffusion

In recent studies of cross-field diffusion (Zimbardo et al. 1995; Zimbardo & Veltri 1995) and analytic (Barghouty & Jokipii 1996) the nonlinearity appears to be an overriding characteristic that points to the non-Gaussian statistics in the problem, i.e., the random-walk of the field lines can no longer be viewed as a brownian-like motion with Gaussian-like statistics but rather (depending

on the strength of the turbulence) as Lévy flights with long-range correlations according to non-Gaussian (Lévy) distributions (e.g., Shlesinger et al. 1995). Another interesting finding in the same studies is that the statistics appear to be consistent with the standard quasi-linear (which is consistent with Gaussian) statistics when the relative turbulence strength is large enough. The studies have shown that for $\delta b \gtrsim .2$ quasi-linear statistics is a valid description of the random-walk of the field lines, whereas for $\delta b \lesssim .2$ the lines tend to exhibit non-Gaussian diffusion and a quasi-linear description appears inadequate. Formal analysis of this behavior attributes this non-Gaussian diffusion to the effects of higher-order correlations and magnetic percolation.

Turning to the scaling of the perpendicular diffusion coefficient with the relative turbulence strength, another important facet in the description of cross-field diffusion, the recent numerical calculation of Gray et al. (1996) has also shown that the quasi-linear scaling, i.e., $D_\perp \propto \delta b^2$ appears adequate for large $\delta b$, consistent with the non-linear calculations alluded to above. In this study the fluctuating field is taken to be composed of a slab-like and a 2-dimensional components, i.e., $\delta\vec{B}(x,y,z) = \delta\vec{B}_{2D}(x,y) + \delta\vec{B}_s(z)$. What was found (for models appropriate for solar wind turbulence) is that the scaling $D_\perp \propto \delta b$ is the correct one for an 80% 2-dimensional + 20% slab turbulence. While the study assumed classical diffusion (i.e., Gaussian statistics was used in the numerical realization of the fluctuating field), it nonetheless points to another interesting feature of the turbulence in that as $\delta b \rightarrow 0$ $D_\perp \rightarrow D_{2D}$, where $D_{2D}$ is the 2D-fluctuations diffusion coefficient as described in the non-perturbative theory of field lines random-walk of Matthaeus et al. (1995). $D_{2D} = \delta B_{2D}\ell_\perp/B_o$, where $\ell_\perp$ is the correlation length of the 2D turbulence, and $D_\perp = 1/2\ \left(D_s + \sqrt{D_s^2 + 4D_{2D}^2}\right)$. This result is to be contrasted with the slab-like coefficient, $D_s = \delta B_s^2 \ell/2B_o^2$, where $\ell$ is the correlation length along the mean field. In short, it appears from this study that in the limit of small turbulence level, the perpendicular diffusion coefficient in 2-d turbulence geometries depends on a length scale other than a parallel one (as the quasi-linear theory would suggest), and appears not to follow the quasi-linear scaling.

For the purpose of this work, it appears from these recent studies of the random walk of the field lines that: (*i*) On the one hand, quasi-linear description may still be appropriate for large relative turbulence strength irrespective of the assumed turbulence geometry (2-d vs. slab) and statistics (non-Gaussian vs. Gaussian). This finding is rather unexpected since in the very development of quasi-linear theory assumptions about small turbulence levels were made for the theory to be applicable and it was further assumed that for large turbulence levels quasi-linear theory may not be applicable due to the method by which the diffusion coefficient was derived, i.e., in the limit of $\delta b^2 \ll 1$ (Jokipii 1966, 1971; Jokipii & Parker 1969a). (*ii*) On the other hand, quasi-linear description of field lines random walk has been shown by these studies to be inadequate for small turbulence levels under the assumption of either Gaussian or non-Gaussian statistics and especially for 2-d turbulence geometries. This is a far reaching conclusion, and its

implications for the heliospheric transport of cosmic rays deserves concerted analytic as well as numerical efforts. One direction is to couple the perpendicular diffusion coefficient to both the relative turbulence strength and an appropriate normal length scale, thus coupling geometry with statistics in a description of the random walk of field lines.

The parameterization of the so-called (effective) anomalous diffusion developed for magnetic turbulence with strong anisotropy (Rosenbluth et al. 1966; Krommes 1978; Rechester & Rosenbluth 1978; Kadomstev & Poguste 1979; Isichenko 1991, 1992) can be used to accomplish such a coupling. In this nonlinear description of the random walk of the field lines, the anisotropy (due to $\kappa_{\parallel} \gg \kappa_{\perp}$ as in solar-wind turbulence geometries) has been shown to significantly enhance the transport across the mean field, i.e., giving rise to an effectively large $\kappa_{\perp}^{*}$.

For a general 2-d magnetic turbulence characterized by correlation lengths along and across the the mean field, $\ell_{\parallel}$ and $\ell_{\perp}$, a Kolmogorov length $\ell_K$ for the exponential divergence of the field lines in the plane normal to the mean field, i.e., $\langle \rho(z) \rangle = \rho_o \exp(z/\ell_K)$, where $\rho(z)$ is as described in Sect. 1.3 and $\rho_o = \rho(z=0)$, and field-lines diffusion coefficient $D_m$ (e.g., $D_{\perp}$ in a 2d-turbulence geometry as alluded to in Sect. 1.1), the effective cross-field diffusion is written

$$\kappa_{\perp}^{*} = \kappa_{\parallel} \frac{D_m}{\ell_K} \left[ \ln \left( \frac{\kappa_{\parallel}}{\kappa_{\perp}} \frac{\ell_{\perp}^2}{\ell_K^2} \right) \right]^{-1} . \tag{23}$$

While the above formulation was developed and applied to magnetic turbulence characterized by large Kubo numbers, $\mathcal{R} \equiv \delta b \, (\ell_{\parallel}/\ell_{\perp}) \gg 1$, i.e., the so-called magnetic percolation regime, its applicability does not appear to be restricted to this regime (Isichenko 1991). We will be applying it in the regime that is characteristic of space and astrophysical plasmas, i.e., $\mathcal{R} \sim \mathcal{O}(\delta b) \lesssim 1$ (Zimbardo et al. 1995; Zimbardo & Veltri 1995; Barghouty & Jokipii 1996).

For $(\kappa_{\parallel}, \ell_{\parallel})$ and $(\kappa_{\perp}, \ell_{\perp})$ the parameterization and values of the (unenhanced) standard description are retained, while noting that $\kappa_{\parallel} \gg \kappa_{\perp}$ and $\ell_{\perp} \simeq \mathcal{R}\ell_{\parallel}$. For $D_m$ we will take advantage of the parameterization of Matthaeus et al. (1995), and for $\ell_K$ we will use a fit to the entropy calculation of Barghouty & Jokipii (1996) wherein $\ell_K \simeq 3\ell_{\parallel}\mathcal{R}^{-1}/2$.

Note that for small $\delta b$ the above parameterization of nonlinear cross-field diffusion takes into account both the effects of the 2d-turbulence geometry on the diffusivity of the field lines (Matthaeus et al. 1995; Gray et al. 1996) via the scaling character of $D_m$ as well as the non-Gaussian statistics of the lines random-walk (Zimbardo et al. 1995; Zimbardo & Veltri 1995; Barghouty & Jokipii 1996) via $\ell_K$. Moreover, when $\delta b$ is large enough the parameterization reduces quite smoothly to the quasi-linear description.

One can first explore the degree to which this nonlinear parameterization deviates from the standard one insofar as the calculated long-term cosmic rays fluxes are concerned throughout the modulation region, with particular emphasis on latitudinal transport and the observed high degree

of isotropy therein, e.g., Ulysses' recent observations (Mckibben et al. 1995a). Also, by calculating time-sensitive cosmic rays fluxes, one is able to focus on short-terms effects of cross-field diffusion, e.g., interactions with CIRs and the current sheet (Jokipii & Kóta 1995; Mckibben et al. 1995b). Finally, and from fits to various observed radial and latitudinal cosmic rays intensity gradients at both low and high latitudes and inner and outer heliospheric locations (Forsyth 1995; Smith et al. 1995; McDonald & Lal 1995), an optimized working set of solar-minimum solar-wind turbulence and HMF parameters can be collected.

### 4.2.2 Modification to the HMF in the Polar Regions

This modification was first suggested by Jokipii & Kóta (1989) to allow the HMF to become larger than the standard Parker's HMF model at high latitudes. It was motivated by recent observations of the mean and fluctuating components of the HMF at high latitudes (e.g., Smith & Bieber, 1991).

The modification simply amounts to adding a $\hat{\phi}$-term of the HMF that is $\propto r$ and implies a non-zero azimuthal component for the HMF as $\theta \rightarrow 0$ (note that the standard Parker's spiral expression suggests that $B_\phi \rightarrow 0$ as $\theta \rightarrow 0$). Smith & Bieber (1991) argued that one could invoke a differential solar rotation as function of $\theta$ to justify this modification.

In principle, such a modification is straightforward to implement in a numerical code. In this code, however, and since we calculate all diffusion and drift terms (which are affected by this modification) analytically so as to minimize the numerical error, implementing the modification is still straightforward but somewhat labor (algebra) intensive. Finally, we note that this particular modification does keep the large-scale HMF a divergence free field, with or without the presence of the sheet, as well as keep $\nabla \cdot \langle \vec{V}_d \rangle = 0$.[7] On the other hand, it is far from clear that this particular modification is of any more consequence than, say, the (still poorly understood) role of the cross-field diffusion in shaping the overall transport picture of galactic cosmic rays in a three-dimensional heliosphere.

### 4.2.3 The Anomalous CR Component

The extension of the code to include the anomalous component is also, in principle, straightforward since the transport equation for this component is essentially identical to Eq. (1) except for source and sink terms to reflect gains and losses caused by ionization. In addition to these terms, one has to assume an initial (or injection) distribution function for the ACR particle in the vicinity of the shock region –around 80 AU– (Jokipii 1990).

[7] Since this new azimuthal term is only a function of $r$, the overall divergence of the HMF is still zero. Also, owing to the identity $\nabla \cdot \nabla \times [\text{vector}] = 0$, and from Eq. (7) it follows that $\langle \vec{V}_d \rangle$ is still divergence free as well.

## 5. Bibliography


Achterberg, A. & Ball, L. 1994, A&A, **285**, 687.

Adams, J. H., Jr. & Lee, J. 1996, Rad. Measur. **26**, 467.

Ananthakrishnan S., Balasubramanian, V., & Janardhan, P. 1995, Sp. Sc. Rev., **72**, 229.

Barge P., Millet, J., & Pellat, R. 1984, ApJ, **284**, 817.

Barghouty, A. F. 1997, *to be submitted to* JGR.

Barghouty, A. F. & Jokipii, J. R. 1996, ApJ, **470**, 858.

Bieber, J. W., Burger, R. A., & Matthaeus, W. H. 1995, 24th ICRC (Rome), **4**, 694.

Blandford, R. & Eichler, D. 1987, Phys. Rep., **154**, 1.

Byrne, G. D. & Hindmarsh, A. C. 1987, J. Comp. Phys., **70**, 1.

Chuvilgin, L. G. & Ptuskin, V. S. 1993, A&A, **279**, 278.

Corbelli, E. & Veltri, P. 1989, ApJ, **340**, 679.

Dendy, R. O., Duffy, P., Gallant, Y. A., & Kirk, J. G. 1995, 24th ICRC (Rome), **3**, 221.

Drury, L. 1983, Rep. Prog. Phys., **46**, 973.

Duffy, P., Kirk, J. G., Gallant, Y. A., & Dendy, R. O. 1995, A&A, **302**, L21.

Evenson, P., Garcia-Munoz, M., Meyer, P., Pyle, K. R., & Simpson, J. A. 1983, ApJ, **275**, L15.

Fisk, L. A. 1971, JGR, **76**, 221.

Fisk, L. A. 1976, JGR, **81**, 4646.

Forsyth, R. J. 1995, Sp. Sc. Rev., **72**, 153.

Garcia-Munoz, M., Pyle, K. R., & Simpson, J. A. 1985, 19th ICRC (La Jolla), **4**, 409.

Gleeson, L. J. & Axford, W. I. 1968, ApJ, **154**, 1011.

Giacalone, J. & Jokipii, J. R. 1994, ApJ, **430**, L137.

Giacalone, J. & Jokipii, J. R. 1996, JGR, **101**, 11095.

Gray, P. C., Pontius D. H., & Matthaeus, W. H. 1996, GRL, **23**, 965.

Hindmarsh, A. C. 1982, in Proc. IMACS Tenth World Congress, Montreal, Canada, 427.

Isichenko, M. B. 1991, Plasma Phys. and Contr. Fusion **33**, 809.

Isichenko, M. B. 1992, Rev. Mod. Phys., **64**, 961.

Jokipii, J. R. 1966, ApJ, **146**, 480.

Jokipii, J. R. 1971, Rev. Geophys. Sp. Phys. **9**, 27.

Jokipii, J. R. 1973, ApJ, **183**, 1029.

Jokipii, J. R. 1987, ApJ, **313**, 842.

Jokipii, J. R., in *Physics of the Outer Heliosphere*, eds. Grzedzielski, S. & Page, D. F. 1990, (Oxford: Pergamon), 169.

Jokipii, J. R. & Davila, J. M. 1981, ApJ, **248**, 1156.

Jokipii, J. R. & Kóta, J. 1989, GRL, **16**, 1.

Jokipii, J. R. & Kóta, J. 1995, Sp. Sc. Rev., **72**, 379.

Jokipii, J. R., Levy, E. H., & Hubbard, W. B. 1976, ApJ, **213**, 861.

Jokipii, J. R. & Morfill, G. 1987, ApJ, **312**, 170.

Jokipii, J. R. & Parker, E. N. 1969a, ApJ, **155**, 777.

Jokipii, J. R. & Parker, E. N. 1969b, ApJ, **155**, 799.

Jokipii, J. R. & Parker, E. N. 1970, ApJ, **160**, 735.

Kadomtsev, B. B. & Pogutse, O. P. 1979, Plasma Phys. and Contr. Nucl. Fusion Res., Proc. 7th Int. Conf. (Vienna) **1**, 649.

Klepach, E. G. 1995, 24th ICRC (Rome), **3**, 112.

Kóta, J. & Jokipii, J. R. 1983, ApJ, **265**, 573; *and reference therein.*

Kóta, J. & Jokipii, J. R. 1995, 24th ICRC (Rome), **4**, 680.

Krommes, J. A. 1978, Prog. Theo. Phys. Suppl. **64**, 137.

Lagage, P. O. & Cesarsky, C. J. 1983, A&A, **125**, 249.

Letaw, J. R., Silberberg, R., & Tsao, C. H. 1984, ApJ Suppl., **56**, 369.

McDonald, F. B. & Lal, N. 1995, 24th ICRC (Rome), **4**, 784.

Mckibben, R. B., Connell, J. J., Lopate, C., Simpson, J. A., & Zhang, M. 1995a, Sp. Sc. Rev., **72**, 367.

Mckibben, R. B., Simpson, J. A., Zhang, M., Bame, S., & Balogh, A. 1995b, Sp. Sc. Rev., **72**, 403.

Matthaeus, W. H., Gray, P. C., Pontius, D. H., Jr., & Bieber, J. W. 1995, Phys. Rev. Lett., **75**, 2136.

Morfill, G. E. & Völk, H. J. 1979, JGR, **84**, 4446.

Perko, J. S. & Fisk, L. A. 1983, GJR, **88**, 9033.

Ptuskin, V. S. 1979, Ap&SS, **61**, 259.

Ptuskin, V. S. 1995, 24th ICRC (Rome), **3**, 128.

Rechester, A. B. & Rosenbluth, M. N. 1978, Phys. Rev. Lett., **40**, 38.

Resonebluth, M. N., Sagdeev, R. Z., Taylor, J. B., & Zaslavskii, G. M. 1966, Nucl. Fusion, **6**, 297.

Schiesser, W. E. 1991, *The Numerical Method of Lines: Integration of Partial Differential Equations*, (San Diego: Academic Press).

Schiesser, W. E. 1996, Appl. Num. Math., **20**, 171.

Shlesinger, M. F., Zaslavsky, G. M., & Frisch, U. 1995, (Eds.), *Lévy Flights and Related Topics in Physics*, (Berlin: Springer).

Skilling, J., McIvor, I., & Holmes, J. A. 1974, MNRAS, **167**, 87.

Smith, C. W. & Bieber, J. W. 1991, ApJ, **370**, 435.

Smith, E. J., Neugebauer, M., Balogh, A., Bame, S. J., Lepping, R. P., & Tsuturani, B. T. 1995, Sp. Sc. Rev., **72**, 165.

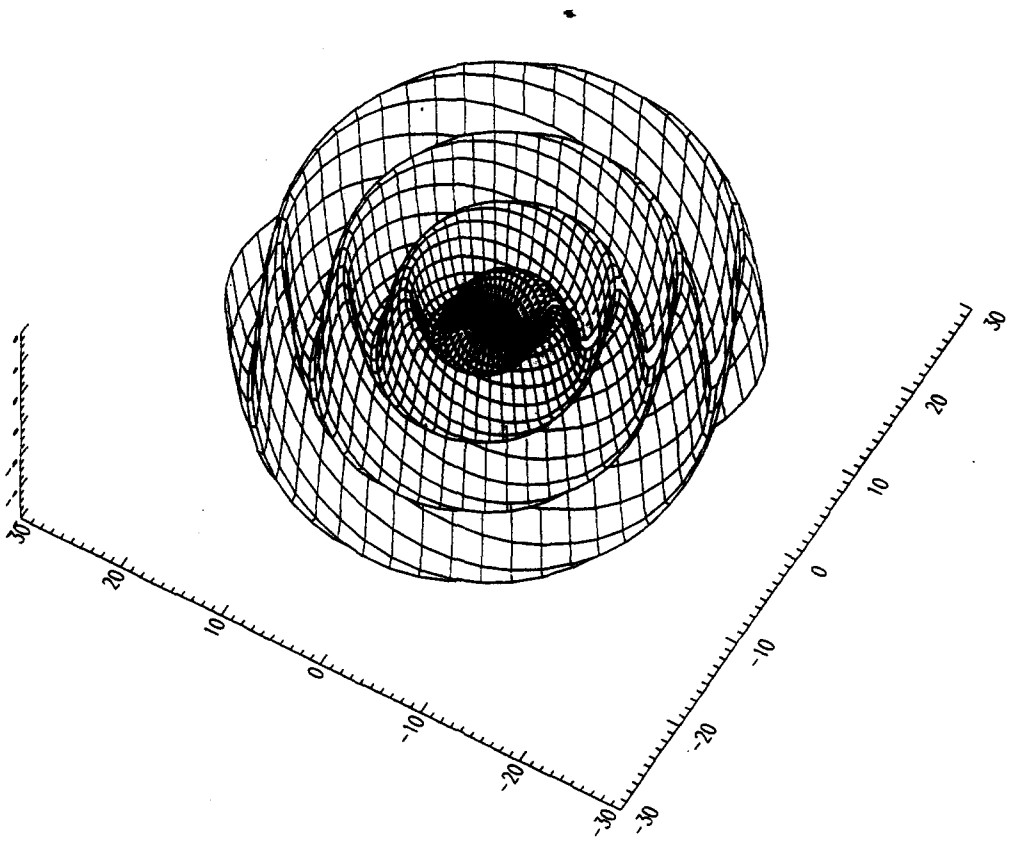
Tylka, A. J., Adams, J. H. Jr., Boberg, P. R., Brownstein, B., Dietrich, W. F., Flueckiger, E. O., Petersen, E. L., Shea, M. A., Smart, D. F., & Smith, E. C. 1997, IEEE Trans. Nucl. Sc., *in press*.

Zimbardo, G. & Veltri, P. 1995, Phys. Rev. E, **51**, 1412.

Zimbardo, G., Veltri, P., Basile, G., & Principato, S. 1995, Phys. Plasmas, **2**, 2653.

Table 1. Salient physical parameters with their values as used in the sample calculations.

| PARAMETER | VALUE | UNITS |
|---|---|---|
| TRANSPORTED PARTICLE Z-NUMBER | 1 | – |
| TRANSPORTED PARTICLE A-NUMBER | 1 | – |
| MAX. PARTICLE KINETIC ENERGY | $1.000 \times 10^{+2}$ | GeV/Nucl. |
| MIN. PARTICLE KINETIC ENERGY | $1.000 \times 10^{-2}$ | GeV/Nucl. |
| INNER HELIOSPHERIC BOUNDARY | $1.000 \times 10^{-2}$ | AU |
| OUTER HELIOSPHERIC BOUNDARY | $1.000 \times 10^{+2}$ | AU |
| LISM-SPECTRUM EXPONENT | $-2.65 \times 10^{0}$ | – |
| STRENGTH OF DIFFUSION TENSOR | $2.500 \times 10^{+3}$ | $AU^2/yr$ |
| PERP. to PARALLEL DIFFUSION | $1.000 \times 10^{-1}$ | – |
| STRENGTH OF DRIFT VECTOR | $1.400 \times 10^{+1}$ | AU/yr |
| STRENGTH OF HMF | $3.322 \times 10^{+1}$ | $\mu$G-$AU^2$ |
| TILT ANGLE OF SHEET @ RMIN | 20 | deg |
| $\delta B$-SPECTRUM EXPONENT | $1.00 \times 10^{0}$ | – |
| SOLAR-WIND SPEED @ 0-LAT. | $9.850 \times 10^{+1}$ | AU/yr |
| SOLAR-WIND HELIOMAG-LAT PAR. | $6.800 \times 10^{-1}$ | – |
| SOLAR ROTATIONAL SPEED | $9.450 \times 10^{+1}$ | rad/yr |
| BOUNDARY CONDITION @ RMIN | $\vec{F} = 0$ | 1/yr |
| GRID SIZE $[r, \theta, \phi, T]$ | $15 \times 15 \times 15 \times 15$ | – |




A perspective of the magnetic current sheet, with $\theta'$ given by Eq. (6), in cartesian coordinates.

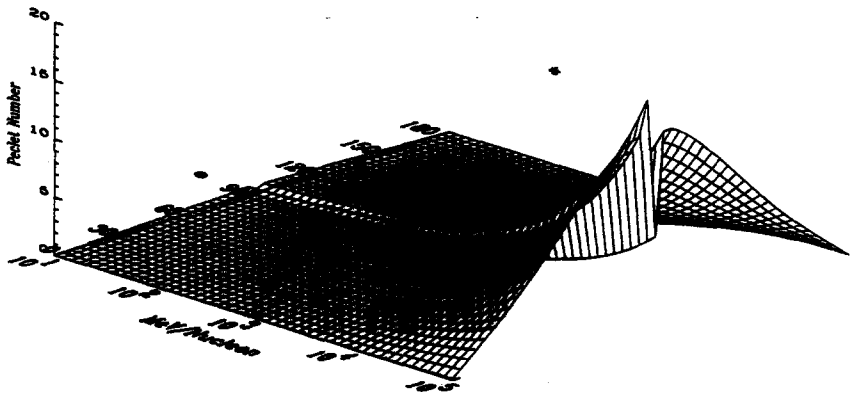



Fig. 1. Péclet spectrum for H as a function of kinetic energy and polar angle at 1 AU, for the $\theta$ groups calculated according to (20).

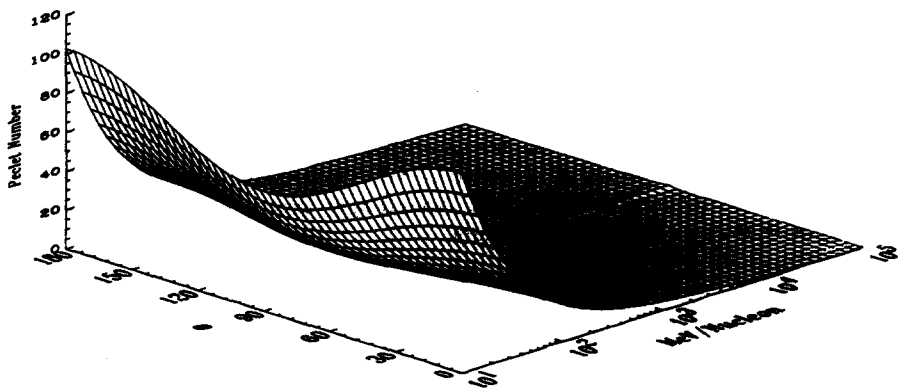



Fig. 2. Péclet spectrum for H as a function of kinetic energy and polar angle at 1 AU, for the $r$ groups calculated according to (21).

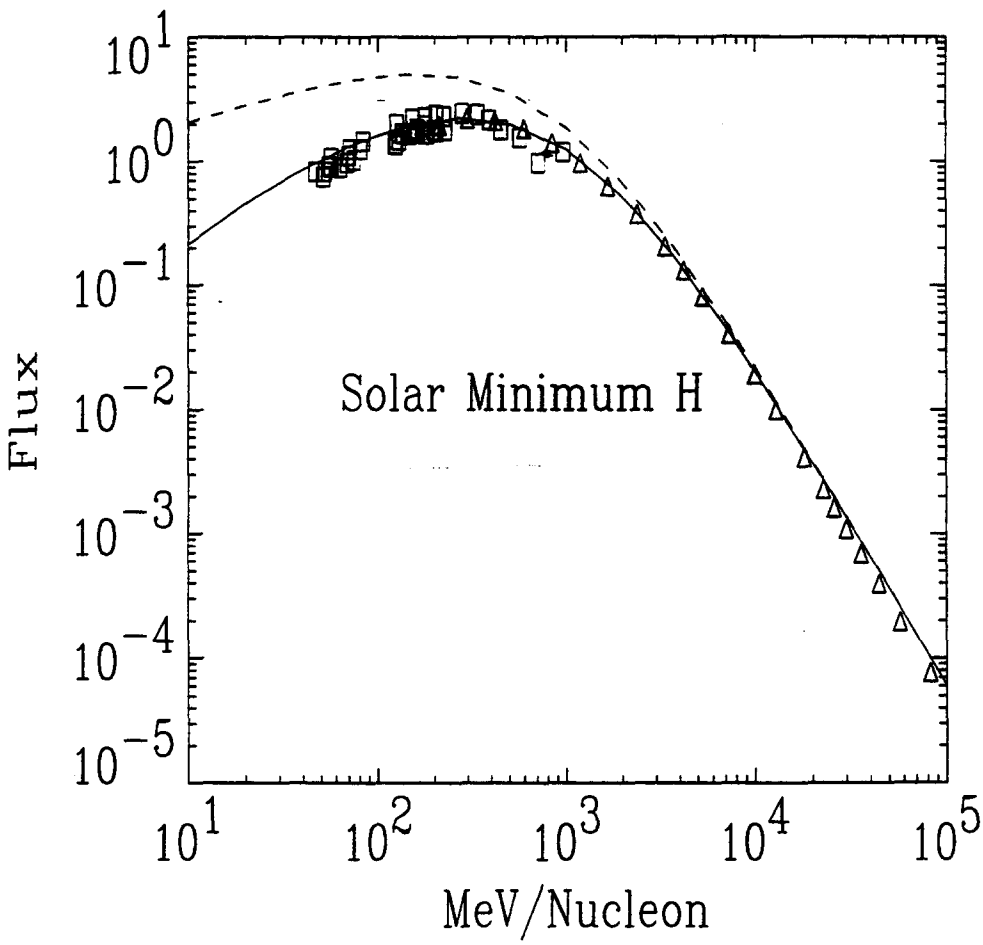



Fig. 3. Model calculations of solar-minimum $\phi$-averaged GCR-H spectrum at $\theta = 90$ deg and 1 AU (solid curve) and at 100 AU (dashed curve). Open triangles are 1987 solar-minimum data while open squares are 1977 data. Note though that the calculations were performed for $qA > 0$ , i.e., 1977 solar-minimum.

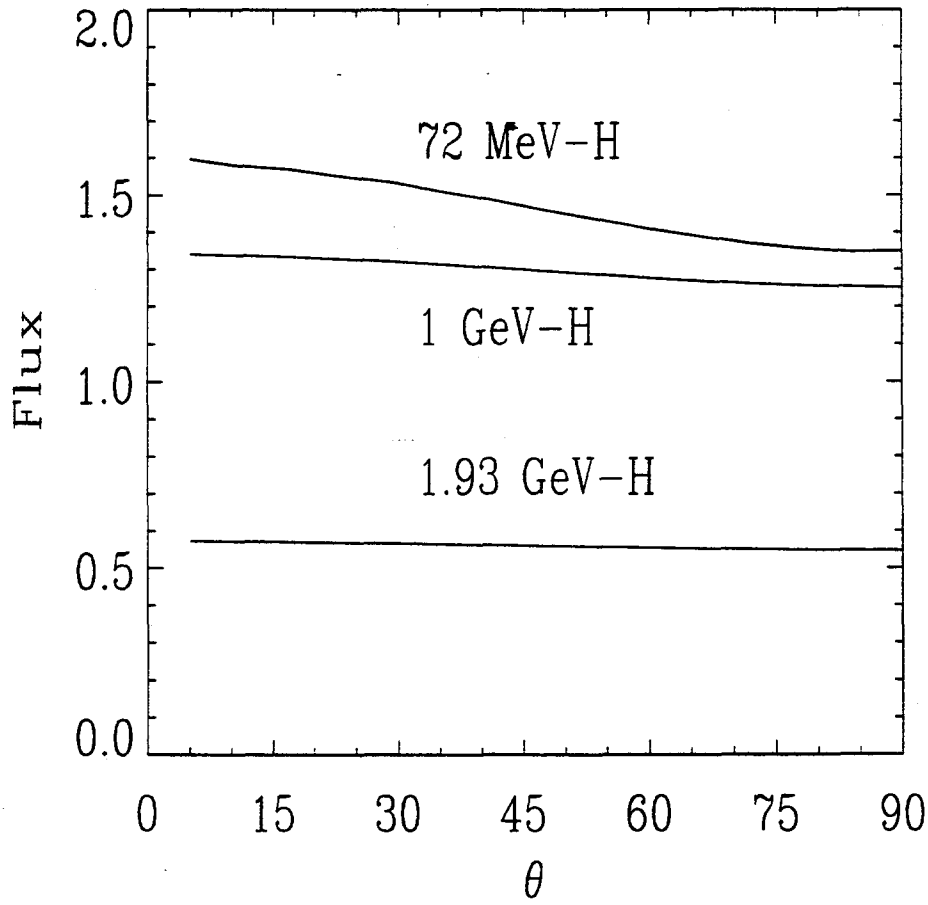



Fig. 4. Calculated $\phi$-averaged solar-minimum H flux at 1 AU as a function of polar angle for three different energies.

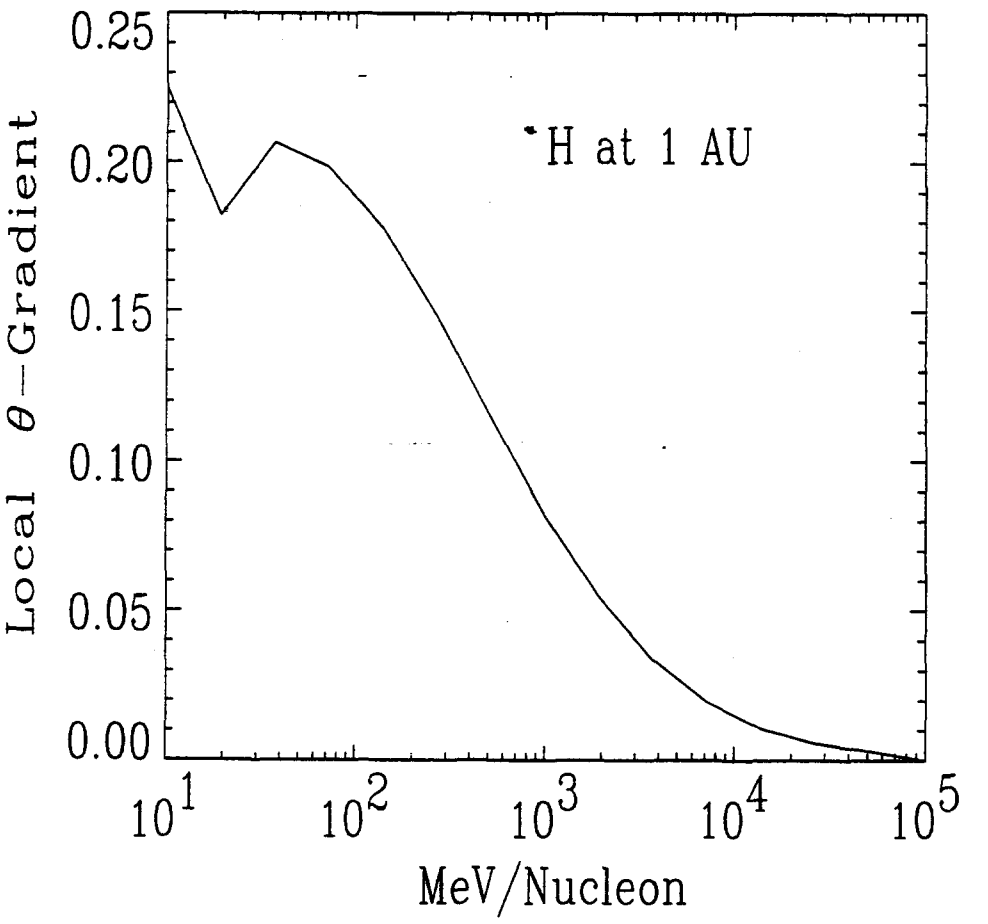



Fig. 5. Calculated local $\theta$-gradient as a function of energy, $\ln[j(5^o)/j(90^o)]/85^o$, in %/deg, at 1 AU.

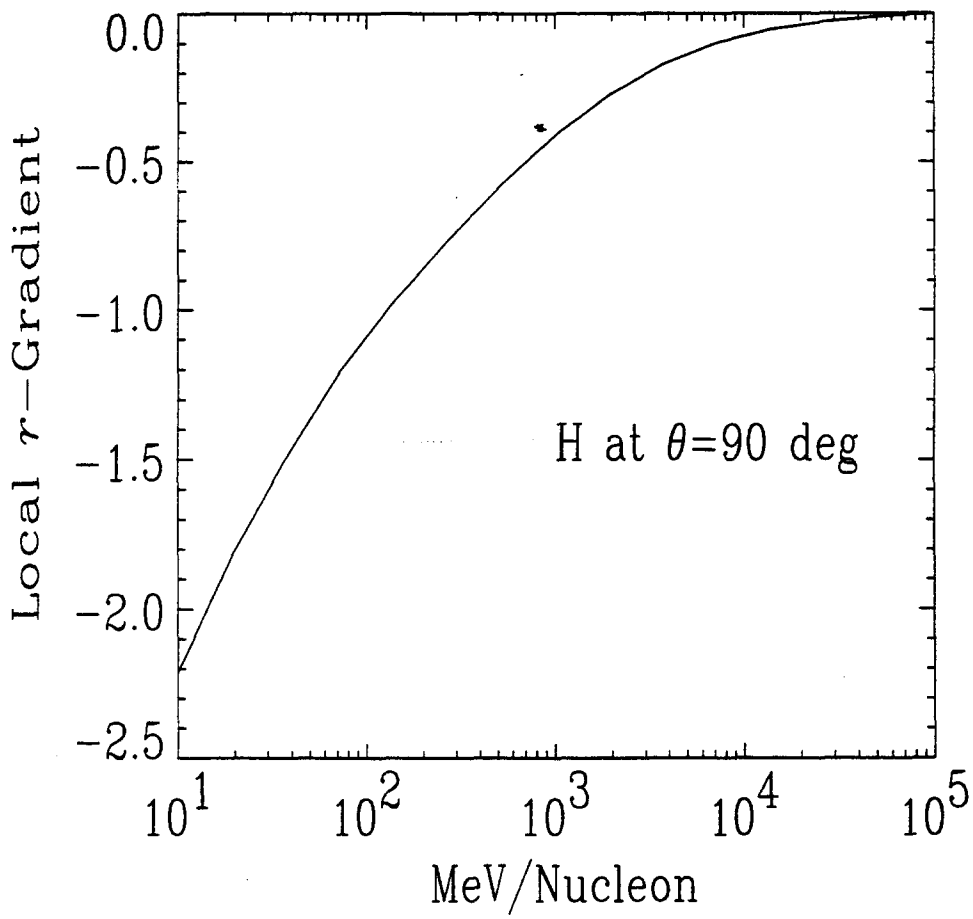
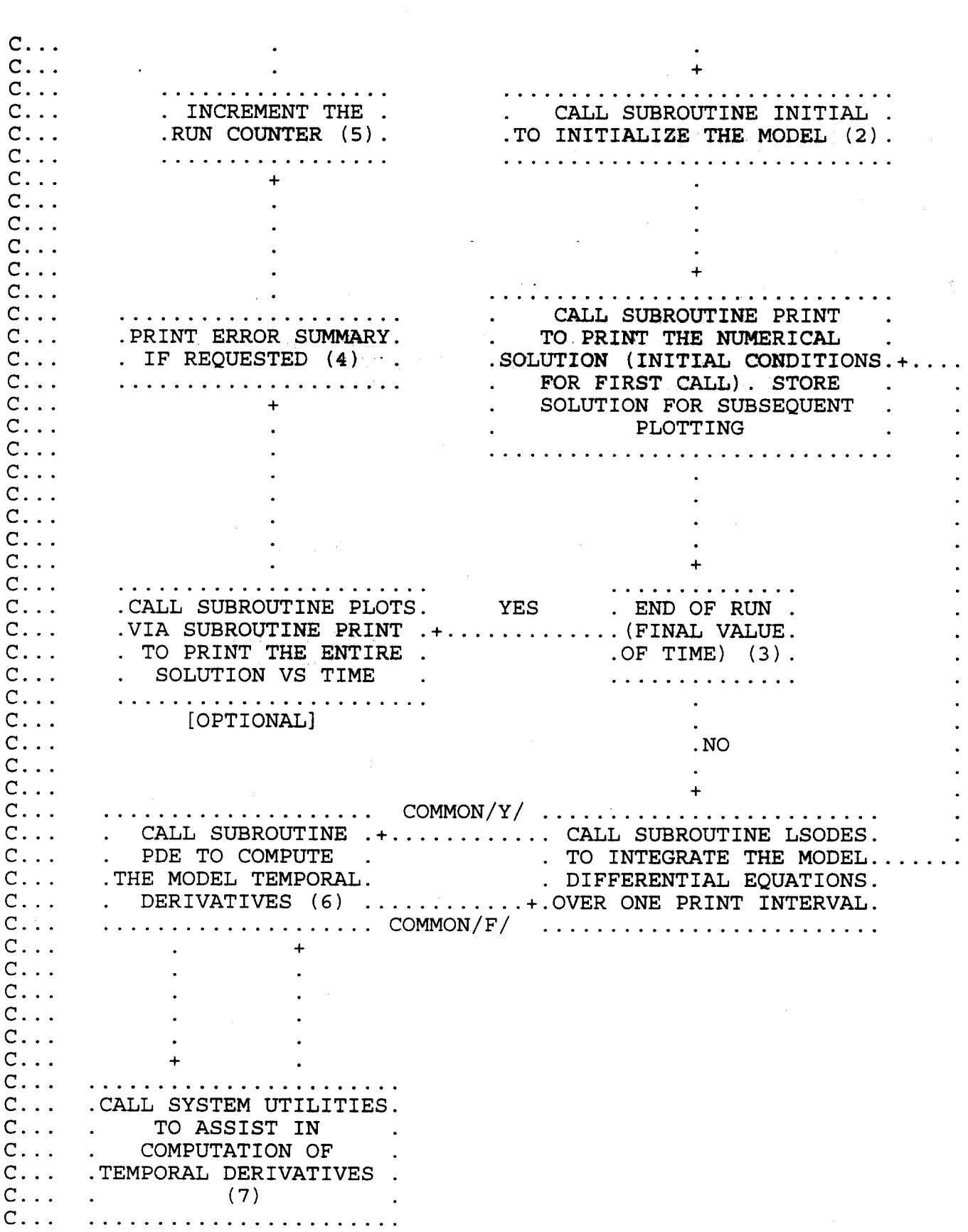



Fig. 6. Calculated local $r$-gradient, $\ln[J(r_o)/J(r_\infty)]/(r_\infty - r_o)$, as a function of energy, in %/AU, at $\theta = 90$ deg.

## Appendix 1. List of Main FORTRAN77 Routines for code PARADIGM

```
      PROGRAM PARADIGM
C...
C.....................................................................
C...
C...  DYNAMIC SOLAR-MODULATION OF GCR IN POLAR SPHERICAL COORDINATES
C...                    MODELLED BY A. F. BARGHOUTY
C...             PHYSICS DEPT., ROANOKE COLLEGE, SALEM, VA 24153
C...        [FOR THE COSMIC-RAY SECTION, NRL, WASHINGTON, DC]
C...             [UNDER NRL GRANT No. N00014-95-1-G037]
C...
C...                              June 1997
C...
C.....................................................................
c...
C...  FLOWCHART OF THE COMPLETE SYSTEM ORGANIZATION AND OPERATION FOR THE
C...   SOLUTION OF THE TRANSPORT EQUATION USING THE NUMERICAL METHOD OF
C...              LINES [cf. Schiesser, W. E. (1991)]
C...
C...                                  ..........................
C...                                  .BEGIN EXECUTION OF MAIN.
C...                                  .     PROGRAM PARADIGM   .
C...                                  ..........................
C...                                             .
C...                                             .
C...                                             .
C...                                             .
C...                                             + (+ DENOTES AN INPUT)
C...                                  ...............
C...                                  .  READ THREE  .
C...                  ................+.DATA LINES (1).
C...                  .               ...............
C...                  .                      .
C...                  .                      .
C...                  .                      .
C...                  .                      .
C...                  .                      +
C...                  .               ............          ......
C...                  .               .END OF RUNS.....+.STOP.
C...                  .               . LINE READ .  YES  ......
C...                  .               ............
C...                  .                      .
C...                  .                      .
C...                  .                      .NO
C...                  .                      .
C...                  .                      +
C...                  .               ............
C...                  .               .PRINT DATA.
C...                  .               .  SUMMARY .
C...                  .               ............
C...                  .                      .
C...                  .                      .
C...                  .                      .
```




EXPLANATORY NOTES FOR THE ABOVE FLOWCHART

(1) LINE 1 - TITLE(20) (READ VIA 900 FORMAT(20A4))

LINE 2 - T0,TF,TP (READ VIA 901 FORMAT(3E10.0))

```
C...      LINE 3 - N,NMAX,NTYPE,NPRINT,IRRTYP,ERROR
C...               (READ VIA 902 FORMAT(4I5,2X,3A1,E10.0))
C...
C...      IF **END OF RUNS** IS ENTERED IN COLUMNS 1 TO 11 OF LINE 1 IN
C...      ANY SET OF THREE DATA LINES, PROGRAM EXECUTION IS TERMINATED
C...      AND LINES 2 AND 3 OF THAT SET ARE NOT REQUIRED.  MULTIPLE
C...      SETS OF DATA LINES MAY BE USED, THREE LINES PER SET.  THE
C...      MAIN PROGRAM WILL READ EACH SET AND EXECUTE A RUN UNTIL AN
C...      **END OF RUNS** LINE IS READ.
C...
C...  (2) SUBROUTINE INITIAL IS CALLED ONCE PER RUN.  THEREFORE DATA
C...      LINES MAY BE READ FROM THIS SUBROUTINE TO DEFINE INITIAL
C...      PARAMETERS OF THE MODEL EQUATIONS FOR EACH RUN.  THE ADDI-
C...      TIONAL DATA LINES WOULD BE PLACED BEHIND THE THREE BASIC
C...      DATA LINES OF (1) ABOVE.
C...
C...  (3) THE END OF RUN CONDITION IS T GE TF WHERE T IS THE FIRST ELE-
C...      MENT IN COMMON/T/ (GENERATED BY MAIN PROGRAM PARADIGM) AND TF
C...      IS READ FROM DATA LINE (2) OF (1) ABOVE.
C...
C...  (4) NPRINT = 1 WILL PRINT A SUMMARY OF THE DEPENDENT VARIABLES IN
C...      COMMON/Y/ FOR WHICH THE ESTIMATED TEMPORAL INTEGRATION (TRUN-
C...      CATION) ERROR EXCEEDED THE MAXIMUM PERMISSIBLE VALUE, ERROR,
C...      (READ FROM DATA LINE (3) OF (1) ABOVE) AT ANY POINT DURING
C...      THE RUN.  IF NPRINT = 0, TEMPORAL INTEGRATION ERRORS WILL NOT
C...      BE REPORTED.
C...
C...  (5) THE RUN COUNTER, SET BY MAIN PROGRAM PARADIGM, IS THE THIRD
C...      ELEMENT IN COMMON/T/ E.G., COMMON/T/T,NFIN,NORUN
C...
C...  (6) THE FUNDAMENTAL LINKAGE IN THIS SYSTEM IS THROUGH COMMON/Y/
C...      WHICH CONTAINS THE MODEL DEPENDENT VARIABLE VECTOR AND COMMON
C...      /F/ WHICH CONTAINS THE VECTOR OF TEMPORAL DERIVATIVES OF THE
C...      DEPENDENT VARIABLE VECTOR.  FOR EXAMPLE, THIS LINKAGE COULD
C...      BE PROGRAMMED AS
C...
C...          COMMON/T/T,NFIN,NORUN/Y/U(NEQ)/F/PUPT(NEQ)
C...
C...      WHERE THE DEPENDENT VARIABLE VECTOR U(NEQ) IS GENERATED BY THE
C...      TEMPORAL INTEGRATOR, SUBROUTINE LSODES, FROM THE DERIVATIVE
C...      VECTOR, PUPT(NEQ), GENERATED BY SUBROUTINE PDE.
C...
C...  MAIN PROGRAM PARADIGM IS THE CALLING PROGRAM FOR A SERIES OF SUB-
C...  ROUTINES WHICH DEFINE AND INTEGRATE THE TEMPORAL DIFFERENTIAL
C...  EQUATIONS.  THE COMPLETE PROGRAM CONSISTS OF THE FOLLOWING
C...  COMPONENTS
C...
C...     (1) MAIN PROGRAM PARADIGM - PERFORMS OVERALL CONTROL OF THE
C...         THE TOTAL PROGRAM.
C...
C...     (2) SUBROUTINE INITIAL - SETS THE INITIAL CONDITIONS FOR THE
C...         TEMPORAL INTEGRATION.
C...
C...     (3) SUBROUTINE PDE - DEFINES THE TEMPORAL DERIVATIVE VECTOR
C...         (PROVIDED BY THE USER). ALSO USES SUPPLIED FUNCTIONS.
C...
C...     (4) SUBROUTINE DIFF - PERFORMS SPATIAL DIFFERENTIATION.
C...
C...     (5) SUBROUTINE LSODES - PERFORMS THE CENTRALIZED TEMPORAL
C...         INTEGRATION.
```

```
C...
      COMMON/T/     T,T0,TF,NSTOP,NORUN
     1      /Y/     Y(10000)
     2      /F/     F(10000)
C...
      COMMON/IO/        NI,         NO
C...
      DIMENSION YV(10000), RWORK(4500000), IWORK(10000)
C...
      CHARACTER*9 REALTIME,REALDATE
      EXTERNAL FCN, JAC
C...
C...  ARRAY FOR THE TITLE (FIRST LINE OF DATA), CHARACTERS  END OF RUNS
      CHARACTER TITLE(20)*4, ENDRUN(3)*4
C...
C...  VARIABLE FOR THE TYPE OF ERROR CRITERION
      CHARACTER*3 ABSREL
C...
C...  DEFINE THE CHARACTERS  END OF RUNS
      DATA ENDRUN/'END ','OF RU','NS  '/
C...
C...  DEFINE THE INPUT/OUTPUT UNIT NUMBERS AND FILES
      NI=5
      NO=6
      CALL TIME(REALTIME)
      CALL DATE(REALDATE)
      OPEN(NI,FILE='INPUT.DAT',STATUS='OLD')
      OPEN(NO,FILE='OUTPUT.DAT',STATUS='NEW')
C...
C...  INITIALIZE THE RUN COUNTER
      NORUN=0
C...
C...  BEGIN A RUN
1     NORUN=NORUN+1
C...
C...  INITIALIZE THE RUN TERMINATION VARIABLE
      NSTOP=0
C...
C...  READ THE FIRST LINE OF DATA
      READ(NI,1000,END=999)(TITLE(I),I=1,20)
C...
C...  TEST FOR  END OF RUNS  IN THE DATA
      DO 2 I=1,3
      IF(TITLE(I).NE.ENDRUN(I))GO TO 3
2     CONTINUE
C...
C...  AN END OF RUNS HAS BEEN READ, SO TERMINATE EXECUTION
999   STOP
C...
C...  READ THE SECOND LINE OF DATA
3     READ(NI,1001,END=999)T0,TF,TP
C...
C...  READ THE THIRD LINE OF DATA
      READ(NI,1002,END=999)NEQN,NMAX,INT,IERROR,ABSREL,ERROR
C...
C...  PRINT A DATA SUMMARY
      WRITE(NO,1003)REALTIME,REALDATE,NORUN,(TITLE(I),I=1,20),
     1                    T0,TF,TP,
     2                    NEQN,NMAX,INT,IERROR,ABSREL,ERROR
C...
```

```
...   INITIALIZE TIME
      T=T0
...
...   SET THE INITIAL CONDITIONS
      CALL INITIAL
C...
...   PRINT THE INITIAL CONDITIONS
      CALL PRINT(NI,NO)
C...
...   SET THE INITIAL CONDITIONS FOR SUBROUTINE LSODES
      DO 5 I=1,NEQN
      YV(I)=Y(I)
5     CONTINUE
...
...   SET THE PARAMETERS FOR SUBROUTINE LSODES
      TV=T0
      ITOL=1
      RTOL=ERROR
      ATOL=ERROR
      LRW=4500000
      LIW=10000
      IOPT=0
      ITASK=1
      ISTATE=1
...
      MF=222
...
...   INITIATE THE INTEGRATION
      TOUT=TV+TP
C...
...   CALL SUBROUTINE LSODES TO COVER ONE PRINT INTERVAL
      CALL LSODES(FCN,NEQN,YV,TV,TOUT,ITOL,RTOL,ATOL,ITASK,ISTATE,
     1            IOPT,RWORK,LRW,IWORK,LIW,JAC,MF)
...
...   PRINT THE SOLUTION
      T=TV
      DO 6 I=1,NEQN
      Y(I)=YV(I)
6     CONTINUE
      CALL PRINT(NI,NO)
...
...   TEST FOR AN ERROR CONDITION
      IF(ISTATE.LT.0)THEN
...
...      PRINT A MESSAGE INDICATING AN ERROR CONDITION
         WRITE(NO,1004)ISTATE
C...
...      GO ON TO THE NEXT RUN
         GO TO 1
      END IF
...
...   CHECK FOR A RUN TERMINATION
      IF(NSTOP.NE.0)GO TO 1
C...
...   THE INTEGRATION HAS PROCEEDED SATISFACTORILY, SO PREPARE FOR THE
...   NEXT INTERVAL IN T.  NOTE THAT THE FOLLOWING CALCULATION OF THE
...   OUTPUT TIME, TOUT, PRODUCES THREE OUTPUT POINTS FOR EACH DECADE
...   IN T
      TOUT=TV*(1.0E+01)**(1.0E+00/3.E+00)
C...
```

```
C...  CHECK FOR THE END OF THE RUN
      IF(TV.LT.(TF-0.5E+00*TP))GO TO 4
C...
C...  THE CURRENT RUN IS COMPLETE, SO PRINT THE COMPUTATIONAL STAT-
C...  ISTICS FOR LSODES AND GO ON TO THE NEXT RUN
      CALL TIME(REALTIME)
      CALL DATE(REALDATE)
      WRITE(NO,8)RWORK(11),IWORK(14),IWORK(11),IWORK(12),IWORK(13),
     1           IWORK(17),IWORK(18),METHOD
8     FORMAT(//10x,'COMPUTATIONAL STATISTICS - TIME INTEGRATION:',/
     1         10x,'=========================================',/
     2         10x,'LAST STEP SIZE                    = ',1PE10.4,/
     3         10x,'LAST ORDER OF THE METHOD          = ',I10,/
     4         10x,'TOTAL NUMBER OF STEPS TAKEN       = ',I10,/
     5         10x,'NUMBER OF FUNCTION EVALUATIONS    = ',I10,/
     6         10x,'NUMBER OF JACOBIAN EVALUATIONS    = ',I10,/
     7         10X,'LENGTH OF ARRAY[RWORK] REQUIRED = ',I10,/
     8         10X,'LENGTH OF ARRAY[IWORK] REQUIRED = ',I10,/
     9         10X,'INTEGRATION METHOD                = ',I10,/
     1         10X,'=========================================='/)
      WRITE(NO,81)NORUN,REALTIME,REALDATE
81    FORMAT(10X,'END OF RUN NO. - ',I2,'    @  ',A8,2X,A9/)
      GO TO 1
C...
C...  ****************************************************************
C...
C...  FORMATS
C...
1000  FORMAT(20A4)
1001  FORMAT(3E10.0)
1002  FORMAT(4I5,2X,A3,E10.0)
1003  FORMAT(/
     1 2X,A8,2X,A9,/,
     1 1X,' RUN NO. - ',I2,2X,20A4,//,
     2 1X,' INITIAL TIME - ',1pE10.3,/,
     3 1X,' FINAL TIME - ',1pE10.3,/,
     4 1X,' PRINT TIME - ',1pE10.3,/,
     5 1X,' NUMBER OF DIFFERENTIAL EQUATIONS - ',I5,/,
     6 1X,' PRINT INTERVAL/MINIMUM INTEGRATION INTERVAL - ',I5,/,
     7 1X,' INTEGRATION ALGORITHM - ',I2,' - LSODES ',/,
     8 1X,' INTEGRATION ERROR MESSAGES - ',I1,/,
     9 1X,' ERROR CRITERION - ',A3,/,
     A 1X,' MAXIMUM INTEGRATION ERROR - ',1PE10.3)
1004  FORMAT(1X,/,' ISTATE = ',I3,/,
     1 ' INDICATING AN INTEGRATION ERROR, SO THE CURRENT RUN'      ,/,
     2 ' IS TERMINATED.  PLEASE REFER TO THE DOCUMENTATION FOR'    ,/,
     3 ' SUBROUTINE',/,25X,'LSODES',/,
     4 ' FOR AN EXPLANATION OF THESE ERROR INDICATORS'             )
      END
      SUBROUTINE FCN(NEQN,TV,YV,YDOT)
C...
C...  SUBROUTINE FCN IS AN INTERFACE ROUTINE BETWEEN SUBROUTINES LSODES
C...  AND DERV
C...
C...  DSS/2 COMMON AREA
      COMMON/T/      T,T0,TF,NSTOP,NORUN
     1       /Y/     Y(1)
     2       /F/     F(1)
C...
C...  VARIABLE DIMENSION THE DEPENDENT AND DERIVATIVE ARRAYS
```

```
      DIMENSION YV(NEQN), YDOT(NEQN)
C...
C...  TRANSFER THE INDEPENDENT VARIABLE, DEPENDENT VARIABLE VECTOR
  ...  FOR USE IN SUBROUTINE DERV
      T=TV
      DO 1 I=1,NEQN
      Y(I)=YV(I)
1     CONTINUE
C...
C...  EVALUATE THE DERIVATIVE VECTOR
      CALL PDE
C...
C...  TRANSFER THE DERIVATIVE VECTOR FOR USE BY SUBROUTINE LSODES
      DO 2 I=1,NEQN
      YDOT(I)=F(I)
2     CONTINUE
      RETURN
      END
C
      SUBROUTINE JAC
  ...
C...  SUBROUTINE JAC IS A DUMMY ROUTINE TO SATISFY THE LOADER (SINCE JAC
C...  IS DECLARED AS AN EXTERNAL IN THE MAIN PROGRAM LSODES).  JAC IS NOT
  ...  ACTUALLY CALLED UNLESS AN OPTION OF LSODE IS SELECTED WHICH REQUIRES
  ...  THE USER TO PROVIDE THE ODE ANALYTICAL JACOBIAN MATRIX.  THIS IS
C...  USUALLY NOT THE CASE SINCE FOR MOST PROBLEMS, THE JACOBIAN MATRIX
C...  IS CALCULATED INTERNALLY BY LSODE USING FINITE DIFFERENCES
      RETURN
      END
C
```

```
C.........................CONSTANTS' FILE.................................
C
C   Constant                    Description
C
C.....................................................................
C.....................................................................
C
C    TILT0        Tilt angle of the magnetic neutral sheet AT the Sun.
C                 TILT0 corresponds to a minimium tilt of 10 deg at
C                 Solar minimum and a maximum tilt of 30 deg at
C                 Solar maximum, i.e., the tilt of the sheet correlates
C                 with the 11-yr sunspot cycle.
C                 ************************************************
C                 *******  AT TIME=0 yr, SOLAR MINIMUM IS ASSUMED  *********
C                 ************************************************
C                 TILT0 is taken here to be 20.0 deg.
C                 in radians;
C
C    B0           Strength of the photospheric magnetic field x solar
C                 radius^2 - corresponds to a field strength of
C                 50. micro-Gauss at Earth's orbit. (Could be positive
C                 or negative depending on polarity and the 22-yr solar
C                 cycle!) Assumed positive over the first 11-yr half
C                 solar cycle and negative over the second.
C                 in (micro-Gauss x AU^2);
C
C    Ws           Solar rotational speed (corresponds to 3.0E-6 rad/s)
C                 in rad/yr;
C
C    Vsw_0        Solar-wind speed (assumed radial of 400 Km/s) at 0-deg
C                 heliomagnetic latitude; Vsw has a heliomagnetic profile
C                 in AU/yr;
C
C    W0           The factor W0 is a phenomenological solar-activity
C                 factor - when W0=0 the solar wind is latitude-indep.;
C                 it is  about 0.68 under solar-minumum conditions for
C                 a velocity profile that increases with heliomagnetic
C                 latitude
C                 dimensionless;
C
C    D0           Strength of local diffusion terms
C                 (corresponds to 1.5E+22 cm^2/s)
C                 in AU^2/yr;
C
C    ETA          Relative strength of normal diffusion term to tangential
C                 term (in local solar-wind frame). ETA is typically << 1
C                 dimensionless;
C
C    V0           Strength of drift-velocity terms
C                 in AU/yr
C
C    E0           amu Rest-mass
C                 in GeV/c^2;
C
C    Z            Atomic number of particle being transported
C                 dimensionless;
C
C    A            Mass number of particle being transported
C                 dimensionless;
C
```

```
C     Rmax        Upper range of radial distance from the SUN, i.e.,
C                 outer Heliospheric boundary
C                 in AUs
C     Rmin        Lower range of radial distance from the SUN, i.e.,
C                 inner Heliospheric boundary - nonzero
C                 in AUs
C     Pmax        Upper range of polar angle -w.r.t. rotational axis of
C                 the SUN
C                 in rad
C     Pmin        Lower range of polar angle -w.r.t. rotational axis of
C                 the SUN
C                 in rad
C     Amax        Upper range of azimuthal angle around the Sun
C                 in rad
C     Amin        Lower range of azimuthal angle around the Sun
C                 in rad
C     Emax        Upper range of particle's kinetic energy
C                 in GeV/nucleon
C     Emin        Lower range of particle's kinetic energy
C                 in GeV/nucleon
C
C     EXPONENT    This is the exponent in total energy of the assumed
C                 LISM density - typically -2.65 to -2.75
C
C     EXP         This is the exponent in the power spectrum of the fluctuating
C                 component of the Heliospheric magnetic field - a purely
C                 Kolmogorov spectrum has EXP=5/3; a Kraichnan spectrum has
C                 EXP=3/2, etc.
C
C     BC          This is the assumed boundary condition at R=RMIN; when
C                 BC='U0' the model assumes U(RMIN)=0. for all times, i.e.,
C                 the density function and hence the intensity is assumed
C                 zero at R=RMIN, and when BC='SF', the model assumes the
C                 streaming flux being zero at R=RMIN, i.e., absorbing Sun,
C                 which satisfies Liouville's theorem.
C
C     c           Speed of light in AU/yr
C.....................................................................
C
C.....Grid:   3-dimensional heliocentric polar spherical
C                   coordinate system) + Energy
C              ( Radial, POLAR,  Azimuthal,  Kinetic Energy)
      PARAMETER( NR=15,  NP=15,  NA=15,      NE=15          )

      CHARACTER*2  BC
C
C.....LISM-spectrum:
      DATA EXPONENT/-2.65/
C
C.....Fluctuating component of HMF:
      DATA EXP/1.000/
C
C.....Regular component of HMF:
      DATA B0,TILT0/33.218,.349/
C
C.....Solar Wind:
      DATA WS,Vsw_0,W0/94.5,84.4,.68/
C
C.....Drift:
```

```
      DATA V0/1.4E+4/
C
C.......Diffusion:
      DATA D0,ETA/2.5E+3,.10/
C
C.......Particle:
      DATA E0,ZNo,ANo/0.938,1.,1./
C
C.......3-D Grid Ranges:
      DATA RMAX,RMIN/100.,.01/
C
      DATA PMAX,PMIN/1.57080,8.72664E-2/
      DATA AMAX,AMIN/3.14159,.0/
C
C.......Energy Range:
      DATA EMAX,EMIN/100.,.01/
C
C.......Boundary condition at R=RMIN:
      DATA BC/'SF'/
C
C.......Physical and mathematical constants:
      DATA c,PI/6.31E+4,3.14159/
C
```

```
 EST RUN(S) FOR THE SOLUTION OF THE JP EQUATION
0.E+00     5.00E+00     1.E-6
 06251000     2     1    REL 1.E-02
 ND OF RUNS
```

```
C.....................COMMON BLOCKS FILE.....................................
C
      COMMON/T/  T, T0, TF, NSTOP, NORUN
     1       /Y/  U(NR,NP,NA,NE),TYME(0:1000000)
     2       /F/  UT(NR,NP,NA,NE)
     3       /S/  UR(NR,NP,NA,NE), URR(NR,NP,NA,NE), URA(NR,NP,NA,NE),
     4            UP(NR,NP,NA,NE), UPP(NR,NP,NA,NE),
     5            UA(NR,NP,NA,NE), UAA(NR,NP,NA,NE),
     6            AD(NR,NP,NA,NE), ADE(NR,NP,NA,NE), ADER(NR,NP,NA,NE),
     7            U0(NR,NP,NA,NE), FLUX_r(NR,NP,NA,NE),
     8            FLUX_p(NR,NP,NA,NE), FLUX_a(NR,NP,NA,NE),
     9            ANISOT_r(NR,NP,NA,NE), ANISOT_p(NR,NP,NA,NE),
     9            ANISOT_a(NR,NP,NA,NE), FLUX(NR,NP,NA,NE)
     1       /C/  RMIN, RMAX, PMIN, PMAX, AMIN, AMAX, EMIN, EMAX,
     2            R(NR), P(NP), A(NA), E(NE), IEO, JEO, KEO, BC
      COMMON/PDE/DIFFUSION,CONVECTION,DRIFT,ADIABATIC,SOURCE,
     1           R_DIFFUSION,A_DIFFUSION,P_DIFFUSION,R_DRIFT,P_DRIFT,
     2           A_DRIFT
```

```
      SUBROUTINE INITIAL
C...
      INCLUDE 'CONSTS.MODEL'
      INCLUDE 'COMMON.MODEL'
C
      common nprint,ip
      COMMON/IO/ NI,NO

C...
      NPRINT=1
...
C...  COMPUTE THE RADIAL, POLAR, AZIMUTHAL, AND ENERGY POSITIONS:
C...  NOTE: RADIAL AND ENERGY POSITIONS ARE LOGARITHMICALLY SPACED!
...
      DELTA_R=(RMAX/RMIN)**(1./(NR-1.))
      IEO=1
      DO I=1,NR
         IF(I.EQ.1) THEN
            R(1)=RMIN
         ELSE
            R(I)=R(I-1)*DELTA_R
         END IF
         IF (R(I).GT.0.75.AND.R(I).LT.1.15) IEO=I
      END DO
...
      JEO=1
      DO J=1,NP
         P(J)=(J-1)*(PMAX-PMIN)/(NP-1)+PMIN
      END DO
C...
      KEO=1
      DO K=1,NA
         A(K)=(K-1)*(AMAX-AMIN)/(NA-1)+AMIN
      END DO
...
      DELTA_E=(EMAX/EMIN)**(1./(NE-1.))
      DO L=1,NE
         IF(L.EQ.1) THEN
            E(1)=EMIN
         ELSE
            E(L)=E(L-1)*DELTA_E
         END IF
      END DO
...
...   SET THE INITIAL CONDITIONS AT TIME=0, I.E., LOCAL-ISM SPECTRUM:
      DO I=1,NR
       DO J=1,NP
        DO K=1,NA
         DO L=1,NE
            IF(I.EQ.NR) THEN
             U0(I,J,K,L)=SPECTRUM(E(L))*VELOCITY(E(L))/(4.*PI)
             U(I,J,K,L)=U0(I,J,K,L)
            ELSE
             U(I,J,K,L)=0.
            END IF
         END DO
        END DO
       END DO
      END DO
C...
```

```
RETURN
END .
```

```
      SUBROUTINE PDE
C...
      LOGICAL EARTH_ORBIT
      INCLUDE 'CONSTS.MODEL'
      INCLUDE 'COMMON.MODEL'
C...
      ic=ic+1   !COUNTER
      TYME(IC)=T
C...
C...  ZERO ALL TERMS;
C...  U is omni-directional, differential CR-intensity function that is
C...  being transported;
C...  [in no. of particles/(m^2-sr-s-GeV/N)]:
      DO I=1,NR
       DO J=1,NP
        DO K=1,NA
         DO L=1,NE
            UT(I,J,K,L)=0.
            UR(I,J,K,L)=0.
            URR(I,J,K,L)=0.
            URA(I,J,K,L)=0.
            UP(I,J,K,L)=0.
            UPP(I,J,K,L)=0.
            UA(I,J,K,L)=0.
            UAA(I,J,K,L)=0.
            AD(I,J,K,L)=0.
            ADE(I,J,K,L)=0.
            ADER(I,J,K,L)=0.
            R_DIFFUSION=0.
            P_DIFFUSION=0.
            A_DIFFUSION=0.
            DIFFUSION=0.
            CONVECTION=0.
            R_DRIFT=0.
            P_DRIFT=0.
            A_DRIFT=0.
            DRIFT=0.
            ADIABATIC=0.
         END DO
        END DO
       END DO
      END DO
C...
C...  SET BOUNDARY CONDITION AT R=RMAX:
      DO J=1,NP
       DO K=1,NA
        DO L=1,NE
          U(NR,J,K,L)=U0(NR,J,K,L)
        END DO
       END DO
      END DO
C...
C...  SET THE BOUNDARY CONDITION AT E=EMAX
      DO I=1,NR
       DO J=1,NP
        DO K=1,NA
           U(I,J,K,NE)=U0(NR,J,K,NE)
        END DO
       END DO
      END DO
```

```
C...  SET BOUNDARY CONDITION AT R=RMIN IF BC='U0':
      IF(BC.EQ.'U0') THEN
C...  NOTE: This BC corresponds to  U(RMIN)=0. for all T!
       DO J=1,NP
        DO K=1,NA
         DO L=1,NE
            U(1,J,K,L)=0.
         END DO
        END DO
       END DO
      END IF
C...
C...  ADIABATIC-COOLING TERM:
      DO I=1,NR
       DO J=1,NP
        DO K=1,NA
         DO L=1,NE
            AD(I,J,K,L)=ALPHA(E(L))*E(L)*U(I,J,K,L)
         END DO
        END DO
       END DO
      END DO
C...
C...  COMPUTE THE FIRST-ORDER DERIVATIVES IN E:
      CALL DSS_4d(LOG(EMIN),LOG(EMAX),NR,NP,NA,NE,4,AD,ADE,0.)
      DO I=1,NR
       DO J=1,NP
        DO K=1,NA
         DO L=1,NE
           ADE(I,J,K,L)=ADE(I,J,K,L)/E(L)
         END DO
        END DO
       END DO
      END DO
C
C...  COMPUTE THE SECOND-ORDER MIXED DERIVATIVES IN E-R IF BC='SF':
      IF(BC.EQ.'SF') THEN
C
       CALL DSS_4d(LOG(RMIN),LOG(RMAX),NR,NP,NA,NE,1,ADE,ADER,0.)
C
C...  This mixed derivative is needed as a boundary condition at R=RMIN
C...  when the "S"treaming "F"lux is assumed zero at R=RMIN, rather than
C...  density U being assumed zero. "SF" BC satisfies Liouville's theorem,
C...  whereas BC "U0" does not!
C
       DO I=1,NR
        DO J=1,NP
         DO K=1,NA
          DO L=1,NE
             ADER(I,J,K,L)=ADER(I,J,K,L)/R(I)
          END DO
         END DO
        END DO
       END DO
      END IF
C...
C...  COMPUTE THE FIRST-ORDER DERIVATIVES IN R FOR DIFFUSION:
      CALL DSS_4d(LOG(RMIN),LOG(RMAX),NR,NP,NA,NE,1,U,UR,0.)
      DO I=1,NR
```

```
      DO J=1,NP
       DO K=1,NA
        DO L=1,NE
          UR(I,J,K,L)=UR(I,J,K,L)/R(I)
        END DO
       END DO
      END DO
     END DO
C...
C...  COMPUTE THE SECOND-ORDER DERIVATIVES IN R FOR DIFFUSION:
      CALL DSS_4d(LOG(RMIN),LOG(RMAX),NR,NP,NA,NE,1,UR,URR,0.)
      DO I=1,NR
       DO J=1,NP
        DO K=1,NA
         DO L=1,NE
           URR(I,J,K,L)=URR(I,J,K,L)/R(I)
         END DO
        END DO
       END DO
      END DO
C...
C...  COMPUTE THE SECOND-ORDER MIXED DERIVATIVES IN R-A FOR DIFFUSION:
      CALL DSS_4d(AMIN,AMAX,NR,NP,NA,NE,3,UR,URA,0.)
C...
C...  COMPUTE THE FIRST-ORDER DERIVATIVES IN R FOR CONVECTION:
      CALL DSS_4d(LOG(RMIN),LOG(RMAX),NR,NP,NA,NE,1,U,UR,-1.)
      DO I=1,NR
       DO J=1,NP
        DO K=1,NA
         DO L=1,NE
           UR(I,J,K,L)=UR(I,J,K,L)/R(I)
         END DO
        END DO
       END DO
      END DO
C...
C...  COMPUTE THE FIRST-ORDER DERIVATIVES IN P FOR DIFFUSION:
      CALL DSS_4d(PMIN,PMAX,NR,NP,NA,NE,2,U,UP,0.)
C
C...  SET BOUNDARY CONDITIONs AT P=PMIN and P=PMAX:
       DO I=1,NR
        DO K=1,NA
         DO L=1,NE
            UP(I,1,K,L)=0.
            UP(I,NP,K,L)=0.
         END DO
        END DO
       END DO
C...
C...  COMPUTE THE SECOND-ORDER DERIVATIVES IN P FOR DIFFUSION:
      CALL DSS_4d(PMIN,PMAX,NR,NP,NA,NE,2,UP,UPP,0.)
C...
C...  COMPUTE THE FIRST-ORDER DERIVATIVES IN P FOR CONVECTION:
      CALL DSS_4d(PMIN,PMAX,NR,NP,NA,NE,2,U,UP,0.)

C...  SET BOUNDARY CONDITIONs AT P=PMIN and P=PMAX:
       DO I=1,NR
        DO K=1,NA
         DO L=1,NE
            UP(I,1,K,L)=0.
```

```
            UP(I,NP,K,L)=0.
          END DO
         END DO
        END DO
C...
C...  COMPUTE THE FIRST-ORDER DERIVATIVES IN A FOR DIFFUSION:
      CALL DSS_4d(AMIN,AMAX,NR,NP,NA,NE,3,U,UA,0.)
C
C...  SET BOUNDARY CONDITIONs AT A=AMIN and A=AMAX:
       DO I=1,NR
        DO J=1,NP
         DO L=1,NE
            UA(I,J,1,L)=0.
            UA(I,J,NA,L)=0.
         END DO
        END DO
       END DO
C...
C...  COMPUTE THE SECOND-ORDER DERIVATIVES IN A FOR DIFFUSION:
      CALL DSS_4d(AMIN,AMAX,NR,NP,NA,NE,3,UA,UAA,0.)
C...
C...  COMPUTE THE FIRST-ORDER DERIVATIVES IN A FOR CONVECTION:
      CALL DSS_4d(AMIN,AMAX,NR,NP,NA,NE,3,U,UA,0.)
C
C...  SET BOUNDARY CONDITIONs AT A=AMIN and A=AMAX:
       DO I=1,NR
        DO J=1,NP
         DO L=1,NE
            UA(I,J,1,L)=0.
            UA(I,J,NA,L)=0.
         END DO
        END DO
       END DO
C...
C...  ASSEMBLE THE PDE:
C...
      DO I=1,NR
       DO J=1,NP
        DO K=1,NA
         DO L=1,NE
C
C...  The collective sum of radial diffusion terms:
           R_DIFFUSION=UR(I,J,K,L)*(2.*Drr(R(I),P(J),A(K),E(L),T)/R(I)+
     +                    Drr_r(R(I),P(J),A(K),E(L),T))+
     +                  URR(I,J,K,L)*Drr(R(I),P(J),A(K),E(L),T)+
     +                  (UA(I,J,K,L)*(Dra(R(I),P(J),A(K),E(L),T)/R(I)+
     +                    Dra_r(R(I),P(J),A(K),E(L),T))+
     +                  URA(I,J,K,L)*Dra(R(I),P(J),A(K),E(L),T))/
     +                  (R(I)*SIN(P(J)))
C
C...  NOTE: URA(I,J,K,L)=UAR(I,J,K,L); the term appears both
C...        in the radial diffusion terms as well as in the
C...        azimuthal diffusion terms.
C
C...  The collective sum of polar diffusion terms:
           P_DIFFUSION=UP(I,J,K,L)/(R(I)**2)*
     +                  (Dpp(R(I),P(J),A(K),E(L),T)*COS(P(J))/
     +                  SIN(P(J))+Dpp_p(R(I),P(J),A(K),E(L),T))+
     +                  UPP(I,J,K,L)/(R(I)**2)
     +                  *Dpp(R(I),P(J),A(K),E(L),T)
```

```
C...  The collective sum of azimuthal diffusion terms:
      A_DIFFUSION=UAA(I,J,K,L)*Daa(R(I),P(J),A(K),E(L),T)/
     +                  (R(I)*SIN(P(J)))**2+
     +                  URA(I,J,K,L)*Dar(R(I),P(J),A(K),E(L),T)/
     +                  (R(I)*SIN(P(J)))

C...  The collective sum of all diffusion terms in 3d:
      DIFFUSION=R_DIFFUSION+P_DIFFUSION+A_DIFFUSION

C...  The convection term due to a radial solar-wind velocity profile:
      CONVECTION=UR(I,J,K,L)*Vsw(P(J))+
     +                  2.*U(I,J,K,L)*Vsw(P(J))/R(I)

C...  The radial, polar, and azimuthal drift terms:
      R_DRIFT=VDR(R(I),P(J),A(K),E(L),T)*UR(I,J,K,L)

      P_DRIFT=VDP(R(I),P(J),A(K),E(L),T)*UP(I,J,K,L)/R(I)
C
      A_DRIFT=VDA(R(I),P(J),A(K),E(L),T)*UA(I,J,K,L)/
     +               (R(I)*SIN(P(J)))

C...  The collective sum of drift terms in 3d:
      DRIFT=R_DRIFT+P_DRIFT+A_DRIFT

C...  The adiabatic cooling term:
      ADIABATIC=(2.*Vsw(P(J))/(3.*R(I)))*ADE(I,J,K,L)

C...  Assumed sources [apart from ISM boundary condition] if any:
      SOURCE=0.


C...  Calculate Streaming Flux Vector:
C.... Radial component of the streaming flux vector:
      FLUX_r(I,J,K,L)=-Drr(R(I),P(J),A(K),E(L),T)*UR(I,J,K,L)-
     +      Dra(R(I),P(J),A(K),E(L),T)*UP(I,J,K,L)/(R(I)*SIN(P(J)))+
     +      VDR(R(I),P(J),A(K),E(L),T)*U(I,J,K,L)+
     +      Vsw(P(J))*(U(I,J,K,L)-(1./3.)*ADE(I,J,K,L))

C.... Polar component of the streaming flux vector:
      FLUX_p(I,J,K,L)=-Dpp(R(I),P(J),A(K),E(L),T)*UP(I,J,K,L)/R(I)+
     +      VDP(R(I),P(J),A(K),E(L),T)*U(I,J,K,L)
C
C.... Azimuthal component of the streaming flux vector:
      FLUX_a(I,J,K,L)=-Dar(R(I),P(J),A(K),E(L),T)*UR(I,J,K,L)-
     +      Daa(R(I),P(J),A(K),E(L),T)*UA(I,J,K,L)/(R(I)*SIN(P(J)))+
     +      VDA(R(I),P(J),A(K),E(L),T)*U(I,J,K,L)

C...  Calculate Anisotropy Vector:
C.... Radial component of the anisotropy vector:
      IF(U(I,J,K,L).NE.0.) THEN
       ANISOT_r(I,J,K,L)=3.*FLUX_r(I,J,K,L)/(4.*PI*U(I,J,K,L))
C
C.... Polar component of the anisotropy vector:
       ANISOT_p(I,J,K,L)=3.*FLUX_p(I,J,K,L)/(4.*PI*U(I,J,K,L))

C.... Azimuthal component of the anisotropy vector:
       ANISOT_a(I,J,K,L)=3.*FLUX_a(I,J,K,L)/(4.*PI*U(I,J,K,L))
      END IF
C
```

```
C
C...  Assume modulation is negligible at heliospheric boundary,
C...  and at the highest energy [100 GeV/Nucl.]:
           IF(I.EQ.1.AND.BC.EQ.'U0') THEN
              U(I,J,K,L)=0.
              UT(I,J,K,L)=0.
           ELSE IF(I.EQ.1.AND.BC.EQ.'SF') THEN
              UT(I,J,K,L)=-Vsw(P(J))*ADER(I,J,K,L)/3.
           ELSE IF(I.EQ.NR) THEN
              U(I,J,K,L)=U0(NR,J,K,L)
              UT(I,J,K,L)=0.
           ELSE IF(L.EQ.NE) THEN
              U(I,J,K,L)=U0(NR,J,K,L)
              UT(I,J,K,L)=0.
           ELSE
C...  JP transport-equation in 3d at this T:
C.....................................................................
           UT(I,J,K,L)=DIFFUSION-CONVECTION-DRIFT+ADIABATIC+SOURCE
C.....................................................................
           END IF
C
              FLUX(I,J,K,L)=U(I,J,K,L)
C
c            CALL WARN(I,J,K,L,IC)
c             CALL DIAG(I,J,K,L,IC)
C
         END DO
        END DO
       END DO
      END DO
C...
      RETURN
      END
```

```
      Gp=W0*SIN(2.*POLAR)/(1.+W0*SIN(PI/2.-POLAR)**2)
C...  Note that Gp=0 if W0=0, i.e., latitude-indep. solar-velocity!
      Gpp=1.+RADIAL*(Ws/Vsw(POLAR))*TILT(TIME)*
     +        COS(AZIMUTH-RADIAL*(Ws/Vsw(POLAR)))*Gp
C
      POLAR0=PI/2.+TILT(TIME)*SIN(AZIMUTH-RADIAL*(Ws/Vsw(POLAR)))

      B0=SIGN(B0,COS(PI*TIME/22.))
      VD0=V0*(2.*BETA*Pc)/(3.*ZNo*B0*(1.+G1**2)**2)*RADIAL
      VD0=VD0*DIRAC(POLAR,POLAR0)
C
      VDR=VDSR(RADIAL,POLAR,AZIMUTH,ENERGY,TIME)
     +       *SHEET(RADIAL,POLAR,AZIMUTH,ENERGY,TIME)+ VD0*G1*Gpp

      RETURN
      END

      FUNCTION VDSP(RADIAL,POLAR,AZIMUTH,ENERGY,TIME)
C...  Polar component of the drift velocity
C...  in a single sector
C
      INCLUDE 'CONSTS.MODEL'

      GAMMA=1.+ENERGY/E0
      BETA=SQRT(1.-GAMMA**-2)
      Pc=ANo*SQRT(ENERGY*(ENERGY+2.*E0))
      G1=RADIAL*(Ws/Vsw(POLAR))*SIN(POLAR)
      G2=RADIAL*(Ws/Vsw(POLAR))*COS(POLAR)
C
      B0=SIGN(B0,COS(PI*TIME/22.))
      VD0=V0*(2.*BETA*Pc)/(3.*ZNo*B0*(1.+G1**2)**2)*RADIAL
C
      VDSP=VD0*G1*(2.+G1**2)

      RETURN
      END

      FUNCTION VDP(RADIAL,POLAR,AZIMUTH,ENERGY,TIME)
C...  POLAR component of the drift velocity
C...  given the neutral magnetic sheet SHEET

      INCLUDE 'CONSTS.MODEL'

      GAMMA=1.+ENERGY/E0
      BETA=SQRT(1.-GAMMA**-2)
      Pc=ANo*SQRT(ENERGY*(ENERGY+2.*E0))
      G1=RADIAL*(Ws/Vsw(POLAR))*SIN(POLAR)
      G2=RADIAL*(Ws/Vsw(POLAR))*COS(POLAR)
C
      POLAR0=PI/2.+TILT(TIME)*SIN(AZIMUTH-RADIAL*(Ws/Vsw(POLAR)))

      B0=SIGN(B0,COS(PI*TIME/22.))
      VD0=V0*(2.*BETA*Pc)/(3.*ZNo*B0*(1.+G1**2)**2)*RADIAL
      VD0=VD0*DIRAC(POLAR,POLAR0)

      VDP=VDSP(RADIAL,POLAR,AZIMUTH,ENERGY,TIME)
     +       *SHEET(RADIAL,POLAR,AZIMUTH,ENERGY,TIME) +
     +      VD0*RADIAL*TILT(TIME)*COS(AZIMUTH-RADIAL*(Ws/Vsw(POLAR)))*
     +      G1*(Ws/Vsw(POLAR))*(G1**-2-1.)
```

```
C
      RETURN
      END
C
      FUNCTION VDSA(RADIAL,POLAR,AZIMUTH,ENERGY,TIME)
C...  Azimuthal component of the drift velocity
C...  in a single sector
C
      INCLUDE 'CONSTS.MODEL'
C
      GAMMA=1.+ENERGY/E0
      BETA=SQRT(1.-GAMMA**-2)
      Pc=ANo*SQRT(ENERGY*(ENERGY+2.*E0))
      G1=RADIAL*(Ws/Vsw(POLAR))*SIN(POLAR)
      G2=RADIAL*(Ws/Vsw(POLAR))*COS(POLAR)
C
      G=W0*SIN(POLAR)**2/(1.+W0*SIN(PI/2.-POLAR)**2)
C...  Note that G=0 if W0=0, i.e., latitude-indep. solar-velocity!
C
      B0=SIGN(B0,COS(PI*TIME/22.))
      VD0=V0*(2.*BETA*Pc)/(3.*ZNo*B0*(1.+G1**2)**2)*RADIAL
C
      VDSA=VD0*G1*G2*(1.+2.*G)
C
      RETURN
      END
C
      FUNCTION VDA(RADIAL,POLAR,AZIMUTH,ENERGY,TIME)
C...  Azimuthal component of the drift velocity
C...  given the neutral magnetic sheet SHEET
C
      INCLUDE 'CONSTS.MODEL'
C
      GAMMA=1.+ENERGY/E0
      BETA=SQRT(1.-GAMMA**-2)
      Pc=ANo*SQRT(ENERGY*(ENERGY+2.*E0))
      G1=RADIAL*(Ws/Vsw(POLAR))*SIN(POLAR)
      G2=RADIAL*(Ws/Vsw(POLAR))*COS(POLAR)
C
      Gp=W0*SIN(2.*POLAR)/(1.+W0*SIN(PI/2.-POLAR)**2)
C...  Note that Gp=0 if W0=0, i.e., latitude-indep. solar-velocity!
      Gpp=1.+RADIAL*(Ws/Vsw(POLAR))* TILT(TIME)*
     +         COS(AZIMUTH-RADIAL*(Ws/Vsw(POLAR)))*Gp

      POLAR0=PI/2.+TILT(TIME)*SIN(AZIMUTH-RADIAL*(Ws/Vsw(POLAR)))
C
      B0=SIGN(B0,COS(PI*TIME/22.))
      VD0=V0*(2.*BETA*Pc)/(3.*ZNo*B0*(1.+G1**2)**2)*RADIAL
      VD0=VD0*DIRAC(POLAR,POLAR0)
C
      VDA=VDSA(RADIAL,POLAR,AZIMUTH,ENERGY,TIME)
     +       *SHEET(RADIAL,POLAR,AZIMUTH,ENERGY,TIME) + VD0*Gpp
C
      RETURN
      END
C
      FUNCTION VDS(RADIAL,POLAR,AZIMUTH,ENERGY,TIME)
C...  Drift speed in a single sector
C
      VDS=SQRT(VDSR(RADIAL,POLAR,AZIMUTH,ENERGY,TIME)**2
```

```
      FUNCTION ALPHA(ENERGY)
C
      INCLUDE 'CONSTS.MODEL'

      ALPHA=(ENERGY+2.*E0)/(ENERGY+E0)

      RETURN
      END

      FUNCTION SPECTRUM(ENERGY)

      INCLUDE 'CONSTS.MODEL'

C...  CHOSEN SO THAT AT 100 GeV, THE LISM UNMODULATED
C...  INTENSITY MATCHES THE OBSERVED INTENSITY, WHERE THE
C...  MODULATION EFFECTS ARE ASSUMED NEGLIGIBLE!

       CONST=1.23*2.*1.E-3/1.              !for p; lism-exp=-2.65
       CONST=1.5852*1.23*2.*1.E-3/19.018   !for He
       CONST=1.5852*1.23*2.*1.E-3/3651.    !for Fe


      SPECTRUM=CONST*(ENERGY+E0)**EXPONENT

      RETURN
      END

      FUNCTION Velocity(ENERGY)
C...  To convert particle density to particle intensity:

      INCLUDE 'CONSTS.MODEL'

      GAMMA=1.+ENERGY/E0
      BETA=SQRT(1.-GAMMA**-2)

      Velocity=BETA*c

      RETURN
      END

      FUNCTION Vsw(POLAR)
C...  Solar-wind velocity profile - assumed radial but heliomagnetic
C...  latitude dependent:

      INCLUDE 'CONSTS.MODEL'

      Vsw=Vsw_0*(1.+W0*SIN(PI/2.-POLAR)**2)

      RETURN
      END

      FUNCTION Br(RADIAL,POLAR,AZIMUTH,ENERGY,TIME)
C...  Radial component of the solar magnetic field
C...  in a single sector

      INCLUDE 'CONSTS.MODEL'

      B0=SIGN(B0,COS(PI*TIME/22.))
      Br=B0/RADIAL**2

C
```

```
      RETURN
      END
C
      FUNCTION Bphi(RADIAL,POLAR,AZIMUTH,ENERGY,TIME)
C...  Azimuthal component of the solar magnetic field
C...  in a single sector
C
      INCLUDE 'CONSTS.MODEL'
C
      B0=SIGN(B0,COS(PI*TIME/22.))
      Bphi=-(B0/RADIAL)*(Ws/Vsw(POLAR))*SIN(POLAR)
C
      RETURN
      END
C
      FUNCTION B(RADIAL,POLAR,AZIMUTH,ENERGY,TIME)
C...  Strength of the solar magnetic field
C...  in a single sector - TIME-INDEPENDENT
C
      B=SQRT(Br(RADIAL,POLAR,AZIMUTH,ENERGY,TIME)**2+
     +       Bphi(RADIAL,POLAR,AZIMUTH,ENERGY,TIME)**2)
C
C...  Note: polar component of B is identically zero!
C
      RETURN
      END
C
      FUNCTION VDSR(RADIAL,POLAR,AZIMUTH,ENERGY,TIME)
C...  Radial component of the drift velocity
C...  in a single sector
C
      INCLUDE 'CONSTS.MODEL'
C
      GAMMA=1.+ENERGY/E0
      BETA=SQRT(1.-GAMMA**-2)
      Pc=ANo*SQRT(ENERGY*(ENERGY+2.*E0))
      G1=RADIAL*(Ws/Vsw(POLAR))*SIN(POLAR)
      G2=RADIAL*(Ws/Vsw(POLAR))*COS(POLAR)
C
      G=W0*SIN(POLAR)**2/(1.+W0*SIN(PI/2.-POLAR)**2)
C...  Note that G=0 if W0=0, i.e., for a latitude-indep. solar velocity!
C
      B0=SIGN(B0,COS(PI*TIME/22.))
      VD0=V0*(2.*BETA*Pc)/(3.*ZNo*B0*(1.+G1**2)**2)*RADIAL
      VDSR=-VD0*G2*(1.+G*(1.-G1**2))
C
      RETURN
      END
C
      FUNCTION VDR(RADIAL,POLAR,AZIMUTH,ENERGY,TIME)
C...  Radial component of the drift velocity
C...  given the neutral magnetic sheet SHEET
C
      INCLUDE 'CONSTS.MODEL'
C
      GAMMA=1.+ENERGY/E0
      BETA=SQRT(1.-GAMMA**-2)
      Pc=ANo*SQRT(ENERGY*(ENERGY+2.*E0))
      G1=RADIAL*(Ws/Vsw(POLAR))*SIN(POLAR)
      G2=RADIAL*(Ws/Vsw(POLAR))*COS(POLAR)
```

```
     +          + VDSP(RADIAL,POLAR,AZIMUTH,ENERGY,TIME)**2
     +          + VDSA(RADIAL,POLAR,AZIMUTH,ENERGY,TIME)**2)

      RETURN
      END

      FUNCTION VD(RADIAL,POLAR,AZIMUTH,ENERGY,TIME)
...   Drift speed given the neutral magnetic sheet SHEET

      VD=SQRT( VDR(RADIAL,POLAR,AZIMUTH,ENERGY,TIME)**2
     +          + VDP(RADIAL,POLAR,AZIMUTH,ENERGY,TIME)**2
     +          + VDA(RADIAL,POLAR,AZIMUTH,ENERGY,TIME)**2)

      RETURN
      END

      FUNCTION Dt(RADIAL,POLAR,AZIMUTH,ENERGY,TIME)
...   Tangential component of the symmetric diffusion tensor
C...  kappa (in the local solar-wind frame)

      INCLUDE 'CONSTS.MODEL'

      GAMMA=1.+ENERGY/E0
      BETA=SQRT(1.-GAMMA**-2)
      Pc=ANo*SQRT(ENERGY*(ENERGY+2.*E0))
      RIGIDITY=Pc/ZNo

...   Rel. strength of local field to that at Earth's orbit:
      BRel=B(1.,PI/2.,AZIMUTH,ENERGY,TIME)/
     +       B(RADIAL,POLAR,AZIMUTH,ENERGY,TIME)

      IF(RADIAL.LE.1.) THEN
         IF (RIGIDITY.LT.1.) THEN
          Dt=D0*(RIGIDITY)**(2.-EXP)*BETA
         ELSE
           Dt=0.3*D0*(RIGIDITY)**2*BETA
         END IF
      ELSE
         IF (RIGIDITY.LT.1.) THEN
          Dt=D0*(RIGIDITY)**(2.-EXP)*BETA*BRel
         ELSE
           Dt=0.3*D0*(RIGIDITY)**2*BETA*BRel
         END IF
      END IF

      RETURN
      END

      FUNCTION Dn(RADIAL,POLAR,AZIMUTH,ENERGY,TIME)
C...  Normal component of the symmetric diffusion tensor
...   kappa (in the local solar-wind frame)

      INCLUDE 'CONSTS.MODEL'

      Dn=ETA*Dt(RADIAL,POLAR,AZIMUTH,ENERGY,TIME)

      RETURN
      END

      FUNCTION Drr(RADIAL,POLAR,AZIMUTH,ENERGY,TIME)
```

```
C...    Kappa_rr component of the symmetric diffusion tensor kappa
C...    (in the heliocentric polar spherical coordinate system)
C
        INCLUDE 'CONSTS.MODEL'
C
C       The angle between the spiral field line and the radial solar-wind
C       direction:
        PSI=ATAN(RADIAL*Ws/Vsw(POLAR))
C
        Drr=(COS(PSI))**2*Dt(RADIAL,POLAR,AZIMUTH,ENERGY,TIME) +
     +      (SIN(PSI))**2*Dn(RADIAL,POLAR,AZIMUTH,ENERGY,TIME)
C
        RETURN
        END
C
        FUNCTION Drr_r(RADIAL,POLAR,AZIMUTH,ENERGY,TIME)
C...    Radial gradient of the kappa_rr component of the symmetric
C...    diffusion tensor kappa
C
        INCLUDE 'CONSTS.MODEL'
C
C       The angle between the spiral field line and the radial solar-wind
C       direction:
        ZETA=RADIAL*Ws/Vsw(POLAR)
        PSI=ATAN(ZETA)
C
        TERM0=2.-(Bphi(RADIAL,POLAR,AZIMUTH,ENERGY,TIME)/
     +            B(RADIAL,POLAR,AZIMUTH,ENERGY,TIME))**2
        TERM1=TERM0*(COS(PSI)**2+ETA*SIN(PSI)**2)/RADIAL
        TERM2=(Ws/Vsw(POLAR))*(ETA-1.)*SIN(2.*PSI)/(1.+ZETA**2)
C
        IF(RADIAL.LE.1.) THEN
          Drr_r=Dt(RADIAL,POLAR,AZIMUTH,ENERGY,TIME)*(0.    +TERM2)
        ELSE
          Drr_r=Dt(RADIAL,POLAR,AZIMUTH,ENERGY,TIME)*(TERM1+TERM2)
        END IF
C
        RETURN
        END
C
        FUNCTION Dpp(RADIAL,POLAR,AZIMUTH,ENERGY,TIME)
C...    Kappa_thetatheta component of the symmetric diffusion tensor
C...    kappa
C
        INCLUDE 'CONSTS.MODEL'
C
C...    Note because kappa is symmetric; Dpr=Drp=Dpa=Dap=0.,
C...    these terms are included as drift velocity terms!
C
        Dpp=Dn(RADIAL,POLAR,AZIMUTH,ENERGY,TIME)
C
        RETURN
        END
C
        FUNCTION Dpp_p(RADIAL,POLAR,AZIMUTH,ENERGY,TIME)
C...    Polar gradient of the kappa_thetethat component of the symmetric
C...    diffusion tensor kappa
C
        INCLUDE 'CONSTS.MODEL'
C
```

```
      B0=SIGN(B0,COS(PI*TIME/22.))
      TERM1=-.5*(B0*(Ws/Vsw(POLAR))/RADIAL)**2*SIN(2.*POLAR)
      TERM2=B(RADIAL,POLAR,AZIMUTH,ENERGY,TIME)**2

      G=W0*SIN(POLAR)**2/(1.+W0*SIN(PI/2.-POLAR)**2)
C...  Note that G=0 if W0=0, i.e., latitude-indep. solar-velocity!
      TERM3=1.+2.*G

      IF(RADIAL.LE.1.) THEN
        Dpp_p=0.
      ELSE
        Dpp_p=Dn(RADIAL,POLAR,AZIMUTH,ENERGY,TIME)*TERM1*TERM3/TERM2
      END IF

      RETURN
      END

      FUNCTION Daa(RADIAL,POLAR,AZIMUTH,ENERGY,TIME)
C...  kappa_phiphi component of the symmetric diffusion tensor
C...  kappa
...   Note: Daa_a [azimuthal gradient of Daa] is identically zero!
C
      INCLUDE 'CONSTS.MODEL'

...   The angle between the spiral field line and the radial solar-wind
C...  direction:
      PSI=ATAN(RADIAL*Ws/Vsw(POLAR))

      Daa=COS(PSI)**2*Dn(RADIAL,POLAR,AZIMUTH,ENERGY,TIME)+
     +       SIN(PSI)**2*Dt(RADIAL,POLAR,AZIMUTH,ENERGY,TIME)

      RETURN
      END

      FUNCTION Dra(RADIAL,POLAR,AZIMUTH,ENERGY,TIME)
C...  kappa_rphi component of the symmetric diffusion tensor
C...  kappa

      INCLUDE 'CONSTS.MODEL'
C
...   The angle between the spiral field line and the radial solar-wind
...   direction:
      PSI=ATAN(RADIAL*Ws/Vsw(POLAR))

      Dra=(Dn(RADIAL,POLAR,AZIMUTH,ENERGY,TIME)-
     +       Dt(RADIAL,POLAR,AZIMUTH,ENERGY,TIME))*
     +       COS(PSI)*SIN(PSI)

      RETURN
      END

      FUNCTION Dra_r(RADIAL,POLAR,AZIMUTH,ENERGY,TIME)
C...  Radial gradient of the kappa_rphi component of the symmetric
C...  diffusion tensor kappa

      INCLUDE 'CONSTS.MODEL'
C
...   The angle between the spiral field line and the radial solar-wind
....   direction:
      ZETA=RADIAL*Ws/Vsw(POLAR)
```

```
      PSI=ATAN(ZETA)
C
      TERM0=2.-(Bphi(RADIAL,POLAR,AZIMUTH,ENERGY,TIME)/
     +          B(RADIAL,POLAR,AZIMUTH,ENERGY,TIME))**2
      TERM1=TERM0*(ETA-1.)*SIN(2.*PSI)/(2.*RADIAL)
      TERM2=(ETA-1.)*(Ws/Vsw(POLAR))*COS(2.*PSI)/(1.+ZETA**2)
C
      IF(RADIAL.LE.1.) THEN
        Dra_r=Dt(RADIAL,POLAR,AZIMUTH,ENERGY,TIME)*(0.    +TERM2)
      ELSE
        Dra_r=Dt(RADIAL,POLAR,AZIMUTH,ENERGY,TIME)*(TERM1+TERM2)
      END IF
C
      RETURN
      END
C
      FUNCTION Dar(RADIAL,POLAR,AZIMUTH,ENERGY,TIME)
C...  kappa_phir component of the symmetric diffusion tensor
C...  kappa
C
      INCLUDE 'CONSTS.MODEL'
C
C...  Note because kappa is symmetric; Dar=Dra
C
      Dar=Dra(RADIAL,POLAR,AZIMUTH,ENERGY,TIME)
C
      RETURN
      END
C
      FUNCTION Dar_r(RADIAL,POLAR,AZIMUTH,ENERGY,TIME)
C...  Radial gradient of the kappa_phir component of the symmetric
C...  diffusion tensor kappa
C
      INCLUDE 'CONSTS.MODEL'
C
C...  Note because kappa is symmetric; Dar_r=Dra_r
C
      Dar_r=Dra_r(RADIAL,POLAR,AZIMUTH,ENERGY,TIME)
C
      RETURN
      END
c
      FUNCTION SHEET(RADIAL,POLAR,AZIMUTH,ENERGY,TIME)
C...  Magnetic current sheet co-rotating with the Sun
C
      INCLUDE 'CONSTS.MODEL'
C
      POLAR0=PI/2.+TILT(TIME)*SIN(AZIMUTH+RADIAL*(Ws/Vsw(POLAR)))
C
      SHEET=1.-2.*STEP(POLAR,POLAR0)
C
      RETURN
      END
C
      FUNCTION TILT(TIME)
C...  Tilt angle of the Magnetic neutral sheet
C...  at the Sun.  TILT is assumed to correlate
C...  with the 11-yr sunspot cycle, i.e.,
C...  TILT is maximum at solar sunspot maximum and
C...  TILT is minimum at solar sunspot minimum.
```

```
      INCLUDE 'CONSTS.MODEL'

      TILT=TILT0*(1.+.5*COS(2.*PI*TIME/11.))

C     Note: With + in above instead of -, TILT would correspond to
      Solar maximum at t=0!
      RETURN
      END

      FUNCTION STEP(X,Y)
C...  Approximated Heaviside step-function:
C...  for large INDEX

      DATA INDEX/31/

      STEP=.5*(1.+TANH(INDEX*(X-Y)))

      RETURN
      END

      FUNCTION DIRAC(X,Y)
C...  Approximated Dirac delta-function:
...   for large INDEX

      DATA INDEX/31/

       DIRAC=1./COSH(INDEX*(X-Y))**2
       DIRAC=exp(-(INDEX*(X-Y))**2)

      RETURN
      END
```

```
      SUBROUTINE DSS_4d(XL,XU,N1,N2,N3,N4,ND,U4D,UX4D,V)
C...
C...  This is a straight-forward extension of DSS036 to 4 independent
C...  variables.  Written by A. F. Barghouty - July 1996.
C...
C...  SUBROUTINE DSS_4d COMPUTES A PARTIAL DERIVATIVE OVER A FOUR-
C...  DIMENSIONAL DOMAIN USING EITHER FIVE-POINT CENTERED OR FIVE-
C...  POINT BIASED UPWIND APPROXIMATIONS.  IT IS INTENDED PRIMARILY
C...  FOR THE NUMERICAL METHOD OF LINES (NMOL) NUMERICAL INTEGRATION
C...  OF PARTIAL DIFFERENTIAL EQUATIONS (PDES) IN THREE DIMENSIONS.
C...
C...  SUBROUTINE DSS_4d IS CALLED IN ESSENTIALLY THE SAME WAY AS
C...  SUBROUTINE DSS036.  THE ONLY DIFFERENCE IS AN ADDITIONAL ARGU-
C...  MENT, N4, TO DEFINE THE NUMBER OF GRID POINTS IN THE FOURTH
C...  DIMENSION.  THE COMMENTS IN DSS036 SHOULD THEREFORE BE USEFUL
C...  IN UNDERSTANDING THE OPERATION OF DSS_4D.  IN PARTICULAR,
C...  DSS036 CALLS SUBROUTINES DSS004 AND DSS020 TO IMPLEMENT THE
C...  FIVE-POINT CENTERED APPROXIMATION AND FIVE-POINT BIASED UPWIND
C...  APPROXIMATION OF THE PARTIAL DERIAVTIVE, RESPECTIVELY.
C...
C...  ARGUMENT LIST
C...
C...     XL          LOWER VALUE OF THE INDEPENDENT VARIABLE FOR WHICH
C...                 THE PARTIAL DERIVATIVE IS TO BE COMPUTED (INPUT)
C...
C...     XU          UPPER VALUE OF THE INDEPENDENT VARIABLE FOR WHICH
C...                 THE PARTIAL DERIVATIVE IS TO BE COMPUTED (INPUT)
C...
C...     N1          NUMBER OF GRID POINTS FOR THE FIRST INDEPENDENT
C...                 VARIABLE (INPUT)
C...
C...     N2          NUMBER OF GRID POINTS FOR THE SECOND INDEPENDENT
C...                 VARIABLE (INPUT)
C...
C...     N3          NUMBER OF GRID POINTS FOR THE THIRD INDEPENDENT
C...                 VARIABLE (INPUT)
C...
C...     N4          NUMBER OF GRID POINTS FOR THE FOURTH INDEPENDENT
C...                 VARIABLE (INPUT)
C...
C...     ND          NUMBER OF THE INDEPENDENT VARIABLE FOR WHICH THE
C...                 PARTIAL DERIVATIVE IS TO BE COMPUTED (INPUT)
C...
C...     U4D         FOUR-DIMENSIONAL ARRAY CONTAINING THE DEPENDENT
C...                 VARIABLE WHICH IS TO BE DIFFERENTIATED WITH RESPECT
C...                 TO INDEPENDENT VARIABLE ND (INPUT)
C...
C...     UX4D        FOUR-DIMENSIONAL ARRAY CONTAINING THE PARTIAL DERI-
C...                 VATIVE OF THE DEPENDENT VARIABLE WITH RESPECT TO
C...                 INDEPENDENT VARIABLE ND (OUTPUT)
C...
C...     V           VARIABLE TO SELECT EITHER THE FIVE-POINT CENTERED
C...                 OR FIVE-POINT BIASED UPWIND APPROXIMATION FOR THE
C...                 PARTIAL DERIVATIVE.  V EQ 0 CALLS THE FIVE-POINT
C...                 CENTERED APPROXIMATION.  V NE 0 CALLS THE FIVE-POINT
C...                 BIASED UPWIND APPROXIMATION (INPUT)
C...
C...  THE FOLLOWING FOUR-DIMENSIONAL ARRAYS CONTAIN THE DEPENDENT
C...  VARIABLE (U4D) AND ITS PARTIAL DERIVATIVE (UX4D)
      DIMENSION   U4D(N1,N2,N3,N4), UX4D(N1,N2,N3,N4)
```

```
C...
C...  THE FOLLOWING ONE-DIMENSIONAL ARRAYS CONTAIN THE DEPENDENT
C...  VARIABLE (U1D) AND ITS PARTIAL DERIVATIVE (UX1D).  IN EACH
C...  CASE, ONE OF THE INDEPENDENT VARIABLES IS CONSTANT AND THE
C...  OTHER TWO INDEPENDENT VARIABLES VARY OVER THEIR TOTAL INTERVALS.
C...  THESE ARRAYS ARE USED FOR TEMPORARY STORAGE IN CALLING THE
C...  ONE-DIMENSIONAL ROUTINES DSS004 AND DSS020.
C...
C...  NOTE THAT THE ARRAYS HAVE ABSOLUTE DIMENSIONS AND MAY THERE-
C...  FORE HAVE TO BE INCREASED IN SIZE.  HOWEVER, WITH A SIZE
C...  OF 51, THE FOUR-DIMENSIONAL PROBLEM COULD HAVE A GRID OF
C...  51 X 51 X 51 X 51 POINTS, THEREBY GENERATING AN APPROXIMATING ODE
C...  SYSTEM WITH A MULTIPLE OF 51 X 51 X 51 X 51 EQUATIONS, DEPENDING ON
C...  THE NUMBER OF SIMULTANEOUS PDES.  THIS IS A VERY LARGE ODE
C...  PROBLEM, AND THEREFORE THE FOLLOWING ABSOLUTE DIMENSIONING
C...  IS CONSIDERED ADEQUATE FOR MOST PROBLEMS.
      DIMENSION      U1D(51),     UX1D(51)
C...
C...  GO TO STATEMENT 2 IF THE PARTIAL DERIVATIVE IS TO BE COMPUTED
C...  WITH RESPECT TO THE SECOND INDEPENDENT VARIABLE
      IF(ND.EQ.2)GO TO 2
C...
C...  GO TO STATEMENT 3 IF THE PARTIAL DERIVATIVE IS TO BE COMPUTED
C...  WITH RESPECT TO THE THIRD INDEPENDENT VARIABLE
      IF(ND.EQ.3)GO TO 3
C...
C...
C...  GO TO STATEMENT 4 IF THE PARTIAL DERIVATIVE IS TO BE COMPUTED
C...  WITH RESPECT TO THE FOURTH INDEPENDENT VARIABLE
      IF(ND.EQ.4)GO TO 4
C...
C...  **********************************************************************
C...
C...  THE PARTIAL DERIVATIVE IS TO BE COMPUTED WITH RESPECT TO THE
C...  FIRST INDEPENDENT VARIABLE DEFINED OVER AN INTERVAL CONSISTING
C...  OF N1 GRID POINTS.  COMPUTE THE PARTIAL DERIVATIVE AT THE N1 X
C...  N2 X N3 X N4 GRID POINTS VIA NESTED DO LOOPS 09, 10, 11, 12 AND 13
      IF(N1.EQ.1) RETURN
      DO 10 J=1,N2
      DO 11 K=1,N3
      DO 09 L=1,N4
C...
C...  TRANSFER THE DEPENDENT VARIABLE IN THE THREE-DIMENSIONAL ARRAY U3D
C...  TO THE ONE-DIMENSIONAL ARRAY U1D SO THAT SUBROUTINES DSS004 AND
C...  DSS020 CAN BE USED TO CALCULATE THE PARTIAL DERIVATIVE
      DO 12 I=1,N1
      U1D(I)=U4D(I,J,K,L)
 2    CONTINUE
C...
C...  IF V EQ 0, A FIVE-POINT CENTERED APPROXIMATION IS USED FOR THE
C...  PARTIAL DERIVATIVE
      IF(V.EQ.0.)CALL DSS004(XL,XU,N1,U1D,UX1D)
C...
C...  IF V NE 0, A FIVE-POINT BIASED UPWIND APPROXIMATION IS USED FOR
C...  THE PARTIAL DERIVATIVE
      IF(V.NE.0.)CALL DSS020(XL,XU,N1,U1D,UX1D,V)
C...
C...  RETURN THE PARTIAL DERIVATIVE IN THE ONE-DIMENSIONAL ARRAY UX1D
C...  TO THE FOUR-DIMENSIONAL ARRAY UX4D
      DO 13 I=1,N1
```

```
      UX4D(I,J,K,L)=UX1D(I)
13    CONTINUE
C...
C...  THE PARTIAL DERIVATIVE AT PARTICULAR VALUES OF THE SECOND,
C...  THIRD AND FOURTH INDEPENDENT VARIABLE HAS BEEN CALCULATED.
C...  REPEAT THE CALCULATION FOR THE OTHER VALUES OF THE SECOND,
C...  THIRD, AND FOURTH INDEPENDENT VARIABLES
09    CONTINUE
11    CONTINUE
10    CONTINUE
C...
C...  THE PARTIAL DERIVATIVE HAS BEEN CALCULATED OVER THE ENTIRE N1 X
C...  N2 X N3 X N4 GRID.  THEREFORE RETURN TO THE CALLING PROGRAM WITH
C...  THE PARTIAL DERIVATIVE IN THE FOUR-DIMENSIONAL ARRAY UX4D
      RETURN
C...
C...  ********************************************************************
C...
C...  THE PARTIAL DERIVATIVE IS TO BE COMPUTED WITH RESPECT TO THE
C...  SECOND INDEPENDENT VARIABLE DEFINED OVER AN INTERVAL CONSISTING
C...  OF N2 GRID POINTS.  COMPUTE THE PARTIAL DERIVATIVE AT THE N1 X
C...  N2 X N3 X N4 GRID POINTS VIA NESTED DO LOOPS 19, 20, 21, 22 AND 23
2     IF(N2.EQ.1) RETURN
      DO 20 I=1,N1
      DO 21 K=1,N3
      DO 19 L=1,N4
C...
C...  TRANSFER THE DEPENDENT VARIABLE IN THE FOUR-DIMENSIONAL ARRAY U4D
C...  TO THE ONE-DIMENSIONAL ARRAY U1D SO THAT SUBROUTINES DSS004 AND
C...  DSS020 CAN BE USED TO CALCULATE THE PARTIAL DERIVATIVE
      DO 22 J=1,N2
      U1D(J)=U4D(I,J,K,L)
22    CONTINUE
C...
C...  IF V EQ 0, A FIVE-POINT CENTERED APPROXIMATION IS USED FOR THE
C...  PARTIAL DERIVATIVE
      IF(V.EQ.0.)CALL DSS004(XL,XU,N2,U1D,UX1D)
C...
C...  IF V NE 0, A FIVE-POINT BIASED UPWIND APPROXIMATION IS USED FOR
C...  THE PARTIAL DERIVATIVE
      IF(V.NE.0.)CALL DSS020(XL,XU,N2,U1D,UX1D,V)
C...
C...  RETURN THE PARTIAL DERIVATIVE IN THE ONE-DIMENSIONAL ARRAY UX1D
C...  TO THE FOUR-DIMENSIONAL ARRAY UX4D
      DO 23 J=1,N2
      UX4D(I,J,K,L)=UX1D(J)
23    CONTINUE
C...
C...  THE PARTIAL DERIVATIVE AT PARTICULAR VALUES OF THE FIRST,
C...  THIRD, AND FOURTH INDEPENDENT VARIABLE HAS BEEN CALCULATED.
C...  REPEAT THE CALCULATION FOR THE OTHER VALUES OF THE FIRST,
C...  THIRD, AND FOURTH INDEPENDENT VARIABLES
19    CONTINUE
21    CONTINUE
20    CONTINUE
C...
C...  THE PARTIAL DERIVATIVE HAS BEEN CALCULATED OVER THE ENTIRE N1 X
C...  N2 X N3 X N4 GRID.  THEREFORE RETURN TO THE CALLING PROGRAM WITH
C...  THE PARTIAL DERIVATIVE IN THE FOUR-DIMENSIONAL ARRAY UX4D
      RETURN
```

```
C...  ***********************************************************************
C...
C...  THE PARTIAL DERIVATIVE IS TO BE COMPUTED WITH RESPECT TO THE
C...  THIRD INDEPENDENT VARIABLE DEFINED OVER AN INTERVAL CONSISTING
C...  OF N3 GRID POINTS.  COMPUTE THE PARTIAL DERIVATIVE AT THE N1 X
C...  N2 X N3 X N4 GRID POINTS VIA NESTED DO LOOPS 29, 30, 31, 32 AND 33
      IF(N3.EQ.1) RETURN
      DO 30 I=1,N1
      DO 31 J=1,N2
      DO 29 L=1,N4
C...
C...  TRANSFER THE DEPENDENT VARIABLE IN THE FOUR-DIMENSIONAL ARRAY U4D
C...  TO THE ONE-DIMENSIONAL ARRAY U1D SO THAT SUBROUTINES DSS004 AND
C...  DSS020 CAN BE USED TO CALCULATE THE PARTIAL DERIVATIVE
      DO 32 K=1,N3
      U1D(K)=U4D(I,J,K,L)
32    CONTINUE
C...
C...  IF V EQ 0, A FIVE-POINT CENTERED APPROXIMATION IS USED FOR THE
C...  PARTIAL DERIVATIVE
      IF(V.EQ.0.)CALL DSS004(XL,XU,N3,U1D,UX1D)
C...
C...  IF V NE 0, A FIVE-POINT BIASED UPWIND APPROXIMATION IS USED FOR
C...  THE PARTIAL DERIVATIVE
      IF(V.NE.0.)CALL DSS020(XL,XU,N3,U1D,UX1D,V)
C...
C...  RETURN THE PARTIAL DERIVATIVE IN THE ONE-DIMENSIONAL ARRAY UX1D
C...  TO THE FOUR-DIMENSIONAL ARRAY UX4D
      DO 33 K=1,N3
      UX4D(I,J,K,L)=UX1D(K)
33    CONTINUE
C...
C...  THE PARTIAL DERIVATIVE AT PARTICULAR VALUES OF THE FIRST,
C...  SECOND, AND FOURTH INDEPENDENT VARIABLE HAS BEEN CALCULATED.
C...  REPEAT THE CALCULATION FOR THE OTHER VALUES OF THE FIRST,
C...  SECOND AND FOURTH INDEPENDENT VARIABLES
29    CONTINUE
31    CONTINUE
30    CONTINUE
C...
C...  THE PARTIAL DERIVATIVE HAS BEEN CALCULATED OVER THE ENTIRE N1 X
C...  N2 X N3 X N4 GRID.  THEREFORE RETURN TO THE CALLING PROGRAM WITH THE
C...  PARTIAL DERIVATIVE IN THE FOUR-DIMENSIONAL ARRAY UX4D
      RETURN
C...  ***********************************************************************
C...
C...  THE PARTIAL DERIVATIVE IS TO BE COMPUTED WITH RESPECT TO THE
C...  FOURTH INDEPENDENT VARIABLE DEFINED OVER AN INTERVAL CONSISTING
C...  OF N4 GRID POINTS.  COMPUTE THE PARTIAL DERIVATIVE AT THE N1 X
C...  N2 X N3 X N4 GRID POINTS VIA NESTED DO LOOPS 40, 41, 42, 43 AND 44
4     IF(N4.EQ.1) RETURN
      DO 40 I=1,N1
      DO 41 J=1,N2
      DO 42 K=1,N3
C...
C...  TRANSFER THE DEPENDENT VARIABLE IN THE FOUR-DIMENSIONAL ARRAY U4D
C...  TO THE ONE-DIMENSIONAL ARRAY U1D SO THAT SUBROUTINES DSS004 AND
```

```
C...  DSS020 CAN BE USED TO CALCULATE THE PARTIAL DERIVATIVE
      DO 43 L=1,N4
      U1D(L)=U4D(I,J,K,L)
43    CONTINUE
C...
C...  IF V EQ 0, A FIVE-POINT CENTERED APPROXIMATION IS USED FOR THE
C...  PARTIAL DERIVATIVE
      IF(V.EQ.0.)CALL DSS004(XL,XU,N4,U1D,UX1D)
C...
C...  IF V NE 0, A FIVE-POINT BIASED UPWIND APPROXIMATION IS USED FOR
C...  THE PARTIAL DERIVATIVE
      IF(V.NE.0.)CALL DSS020(XL,XU,N4,U1D,UX1D,V)
C...
C...  RETURN THE PARTIAL DERIVATIVE IN THE ONE-DIMENSIONAL ARRAY UX1D
C...  TO THE FOUR-DIMENSIONAL ARRAY UX4D
      DO 44 L=1,N4
      UX4D(I,J,K,L)=UX1D(L)
44    CONTINUE
C...
C...  THE PARTIAL DERIVATIVE AT PARTICULAR VALUES OF THE FIRST,
C...  SECOND, AND THIRD INDEPENDENT VARIABLE HAS BEEN CALCULATED.
C...  REPEAT THE CALCULATION FOR THE OTHER VALUES OF THE FIRST,
C...  SECOND AND THIRD INDEPENDENT VARIABLES
42    CONTINUE
41    CONTINUE
40    CONTINUE
C...
C...  THE PARTIAL DERIVATIVE HAS BEEN CALCULATED OVER THE ENTIRE N1 X
C...  N2 X N3 X N4 GRID.  THEREFORE RETURN TO THE CALLING PROGRAM WITH THE
C...  PARTIAL DERIVATIVE IN THE FOUR-DIMENSIONAL ARRAY UX4D
      RETURN
      END
C
      SUBROUTINE DSS004(XL,XU,N,U,UX)
C...
C...  SUBROUTINE DSS004 COMPUTES THE FIRST DERIVATIVE, U , OF A
C...                                                    X
C...  VARIABLE U OVER THE SPATIAL DOMAIN XL LE X LE XU FROM CLASSICAL
C...  FIVE-POINT, FOURTH-ORDER FINITE DIFFERENCE APPROXIMATIONS
C...
C...  ARGUMENT LIST
C...
C...     XL       LOWER BOUNDARY VALUE OF X (INPUT)
C...
C...     XU       UPPER BOUNDARY VALUE OF X (INPUT)
C...
C...     N        NUMBER OF GRID POINTS IN THE X DOMAIN INCLUDING THE
C...              BOUNDARY POINTS (INPUT)
C...
C...     U        ONE-DIMENSIONAL ARRAY CONTAINING THE VALUES OF U AT
C...              THE N GRID POINT POINTS FOR WHICH THE DERIVATIVE IS
C...              TO BE COMPUTED (INPUT)
C...
C...     UX       ONE-DIMENSIONAL ARRAY CONTAINING THE NUMERICAL
C...              VALUES OF THE DERIVATIVES OF U AT THE N GRID POINTS
C...              (OUTPUT)
C...
C...  THE MATHEMATICAL DETAILS OF THE FOLLOWING TAYLOR SERIES (OR
C...  POLYNOMIALS) ARE GIVEN IN SUBROUTINE DSS002.
C...
```

C...  FIVE-POINT FORMULAS
C...
C...     (1)  LEFT END, POINT I = 1
C...
C...                                                2           3           4
C...  A(U2 = U1 + U1   ( DX) + U1   ( DX)  + U1   ( DX)  + U1   ( DX)
C...               X   1F      2X  2F       3X  3F       4X  4F
C...
C...                        5           6           7
C...          + U1   ( DX)  + U1   ( DX)  + U1   ( DX)  + ...)
C...              5X  5F      6X  6F       7X  7F
C...
C...                                                2           3           4
C...  B(U3 = U1 + U1   (2DX) + U1   (2DX)  + U1   (2DX)  + U1   (2DX)
C...               X   1F      2X  2F       3X  3F       4X  4F
C...
C...                        5           6           7
C...          + U1   (2DX)  + U1   (2DX)  + U1   (2DX)  + ...)
C...              5X  5F      6X  6F       7X  7F
C...
C...                                                2           3           4
C...  C(U4 = U1 + U1   (3DX) + U1   (3DX)  + U1   (3DX)  + U1   (3DX)
C...               X   1F      2X  2F       3X  3F       4X  4F
C...
C...                        5           6           7
C...          + U1   (3DX)  + U1   (3DX)  + U1   (3DX)  + ...)
C...              5X  5F      6X  6F       7X  7F
C...
C...                                                2           3           4
C...  D(U5 = U1 + U1   (4DX) + U1   (4DX)  + U1   (4DX)  + U1   (4DX)
C...               X   1F      2X  2F       3X  3F       4X  4F
C...
C...                        5           6           7
C...          + U1   (4DX)  + U1   (4DX)  + U1   (4DX)  + ...)
C...              5X  5F      6X  6F       7X  7F
C...
C...  CONSTANTS A, B, C AND D ARE SELECTED SO THAT THE COEFFICIENTS
C...  OF THE U1  TERMS SUM TO ONE AND THE COEFFICIENTS OF THE U1   ,
C...           X                                                  2X
C...  U1   AND U1   TERMS SUM TO ZERO
C...    3X       4X
C...
C...  A +   2B +   3C +   4D = 1
C...
C...  A +   4B +   9C +  16D = 0
C...
C...  A +   8B +  27C +  64D = 0
C...
C...  A +  16B +  81C + 256D = 0
C...
C...  SIMULTANEOUS SOLUTION FOR A, B, C AND D FOLLOWED BY THE SOLU-
C...  TION OF THE PRECEDING TAYLOR SERIES, TRUNCATED AFTER THE U
C...                                                             4X
C...  TERMS, FOR U1  GIVES THE FOLLOWING FIVE-POINT APPROXIMATION
C...             X
C...                                                          4
C...  U1   = (1/12DX)(-25U1 + 48U2 - 36U3 + 16U4 - 3U5) + O(DX )    (1)
C...    X
C...
C...     (2)  INTERIOR POINT, I = 2

C...

C...  $$A(U1 = U2 + U2_X(-DX) + U2_{1F}\frac{(-DX)^2}{2!} + U2_{2X}\frac{(-DX)^3}{3!} + U2_{3F}\frac{(-DX)^4}{4!} + U2_{4X}\frac{(-DX)^5}{5!} + U2_{5F}\frac{(-DX)^6}{6!} + U2_{6X}\frac{(-DX)^7}{7!} + U2_{7F}\frac{(-DX)^8}{8!} + \ldots)$$

C...

C...  $$B(U3 = U2 + U2_X(DX) + U2_{1F}\frac{(DX)^2}{2!} + U2_{2X}\frac{(DX)^3}{3!} + U2_{3F}\frac{(DX)^4}{4!} + U2_{4X}\frac{(DX)^5}{5!} + U2_{5F}\frac{(DX)^6}{6!} + U2_{6X}\frac{(DX)^7}{7!} + U2_{7F}\frac{(DX)^8}{8!} + \ldots)$$

C...

C...  $$C(U4 = U2 + U2_X(2DX) + U2_{1F}\frac{(2DX)^2}{2!} + U2_{2X}\frac{(2DX)^3}{3!} + U2_{3F}\frac{(2DX)^4}{4!} + U2_{4X}\frac{(2DX)^5}{5!} + U2_{5F}\frac{(2DX)^6}{6!} + U2_{6X}\frac{(2DX)^7}{7!} + U2_{7F}\frac{(2DX)^8}{8!} + \ldots)$$

C...

C...  $$D(U5 = U2 + U2_X(3DX) + U2_{1F}\frac{(3DX)^2}{2!} + U2_{2X}\frac{(3DX)^3}{3!} + U2_{3F}\frac{(3DX)^4}{4!} + U2_{4X}\frac{(3DX)^5}{5!} + U2_{5F}\frac{(3DX)^6}{6!} + U2_{6X}\frac{(3DX)^7}{7!} + U2_{7F}\frac{(3DX)^8}{8!} + \ldots)$$

C...

C...  $$-A + B + 2C + 3D = 1$$

C...

C...  $$A + B + 4C + 9D = 0$$

C...

C...  $$-A + B + 8C + 27D = 0$$

C...

C...  $$A + B + 16C + 81D = 0$$

C...

C...  SIMULTANEOUS SOLUTION FOR A, B, C AND D FOLLOWED BY THE SOLU-
C...  TION OF THE PRECEDING TAYLOR SERIES, TRUNCATED AFTER THE $U_{4X}$
C...  TERMS, FOR $U1_X$ GIVES THE FOLLOWING FIVE-POINT APPROXIMATION
C...

C...  $$U2_X = (1/12DX)(-3U1 - 10U2 + 18U3 - 6U4 + U5) + O(DX^4) \qquad (2)$$

C...

C...  (3) INTERIOR POINT I, I NE 2, N-1

C...

C...  $$A(UI-2 = UI + UI_X(-2DX) + UI_{1F}\frac{(-2DX)^2}{2!} + UI_{2X}\frac{(-2DX)^3}{3!} + UI_{3F}\frac{(-2DX)^4}{4!} + UI_{4X}\frac{(-2DX)^5}{5!} + UI_{5F}\frac{(-2DX)^6}{6!} + UI_{6X}\frac{(-2DX)^7}{7!} + \ldots)$$

C...

$$B(UI-1 = UI + UI_X^{1F}(-DX) + UI_{2X}^{2F}(-DX)^2 + UI_{3X}^{3F}(-DX)^3 + UI_{4X}^{4F}(-DX)^4 + UI_{5X}^{5F}(-DX)^5 + UI_{6X}^{6F}(-DX)^6 + \ldots)$$

$$C(UI+1 = UI + UI_X^{1F}(DX) + UI_{2X}^{2F}(DX)^2 + UI_{3X}^{3F}(DX)^3 + UI_{4X}^{4F}(DX)^4 + UI_{5X}^{5F}(DX)^5 + UI_{6X}^{6F}(DX)^6 + \ldots)$$

$$D(UI+2 = UI + UI_X^{1F}(2DX) + UI_{2X}^{2F}(2DX)^2 + UI_{3X}^{3F}(2DX)^3 + UI_{4X}^{4F}(2DX)^4 + UI_{5X}^{5F}(2DX)^5 + UI_{6X}^{6F}(2DX)^6 + \ldots)$$

$$-2A - B + C + 2D = 1$$

$$4A + B + C + 4D = 0$$

$$-8A - B + C + 8D = 0$$

$$16A + B + C + 16D = 0$$

SIMULTANEOUS SOLUTION FOR A, B, C AND D FOLLOWED BY THE SOLUTION OF THE PRECEDING TAYLOR SERIES, TRUNCATED AFTER THE $U_X^{4X}$ TERMS, FOR $U1_X$ GIVES THE FOLLOWING FIVE-POINT APPROXIMATION

$$UI_X = (1/12DX)(UI-2 - 8UI-1 + 0UI + 8UI+1 - UI+2) + O(DX^4) \quad (3)$$

(4) INTERIOR POINT, I = N-1

$$A(UN-4 = UN-1 + UN-1_X^{1F}(-3DX) + UN-1_{2X}^{2F}(-3DX)^2 + UN-1_{3X}^{3F}(-3DX)^3 + UN-1_{4X}^{4F}(-3DX)^4 + UN-1_{5X}^{5F}(-3DX)^5 + UN-1_{6X}^{6F}(-3DX)^6 + \ldots$$

$$B(UN-3 = UN-1 + UN-1_X^{1F}(-2DX) + UN-1_{2X}^{2F}(-2DX)^2 + UN-1_{3X}^{3F}(-2DX)^3 + UN-1_{4X}^{4F}(-2DX)^4 + UN-1_{5X}^{5F}(-2DX)^5 + UN-1_{6X}^{6F}(-2DX)^6 + \ldots$$

C... C(UN-2 = UN-1 + UN-1 ( -DX) + UN-1 (- -X) + UN-1 ( -DX)
C... X 1F 2X 2F 3X 3F
C...
C... 4 5 6
C... + UN-1 ( -DX) + UN-1 ( -DX) + UN-1 ( -DX) + ...
C... 4X 4F 5X 5F 6X 6F
C...
C... 2 3
C... D(UN = UN-1 + UN-1 ( DX) + UN-1 ( DX) + UN-1 ( DX)
C... X 1F 2X 2F 3X 3F
C...
C... 4 5 6
C... + UN-1 ( DX) + UN-1 ( DX) + UN-1 ( DX) + ...
C... 4X 4F 5X 5F 6X 6F
C...
C... -3A - 2B - C + D = 1
C...
C... 9A + 4B + C + D = 0
C...
C... -27A - 8B - C + D = 0
C...
C... 81A + 16B + C + D = 0
C...
C... SIMULTANEOUS SOLUTION FOR A, B, C AND D FOLLOWED BY THE SOLU-
C... TION OF THE PRECEDING TAYLOR SERIES, TRUNCATED AFTER THE U
C... 4X
C... TERMS, FOR U1 GIVES THE FOLLOWING FIVE-POINT APPROXIMATION
C... X
C... 4
C... UN-1 = (1/12DX)(-UN-4 + 6UN-3 - 18UN-2 + 10UN-1 + 3UN) + O(DX )
C... X
C... (4)
C...
C... (5) RIGHT END, POINT I = N
C...
C... 2 3
C... A(UN-4 = UN + UN (-4DX) + UN (-4DX) + UN (-4DX)
C... X 1F 2X 2F 3X 3F
C...
C... 4 5 6
C... + UN (-4DX) + UN (-4DX) + UN (-4DX) + ...)
C... 4X 4F 5X 5F 6X 6F
C...
C... 2 3
C... B(UN-3 = UN + UN (-3DX) + UN (-3DX) + UN (-3DX)
C... X 1F 2X 2F 3X 3F
C...
C... 4 5 6
C... + UN (-3DX) + UN (-3DX) + UN (-3DX) + ...)
C... 4X 4F 5X 5F 6X 6F
C...
C... 2 3
C... C(UN-2 = UN + UN (-2DX) + UN (-2DX) + UN (-2DX)
C... X 1F 2X 2F 3X 3F
C...
C... 4 5 6
C... + UN (-2DX) + UN (-2DX) + UN (-2DX) + ...)
C... 4X 4F 5X 5F 6X 6F
C...
C... 2 3

```
C...  D(UN-1 = UN + UN ( -DX)  + UN  ( -DX)  + UN  ( -DX)
C...     .            X   1F       2X   2F       3X   3F
C...
C...                              4             5             6
C...           + UN  ( -DX)  + UN  ( -DX)  + UN  ( -DX)  + ...)
C...              4X   4F       5X   5F       6X   6F
C...
C...   -4A -  3B -  2C -   D = 1
C...
C...   16A +  9B +  4C +   D = 0
C...
C...  -64A - 27B -  8C -   D = 0
C...
C...  256A + 81B + 16C +   D = 0
C...
C...  SIMULTANEOUS SOLUTION FOR A, B, C AND D FOLLOWED BY THE SOLU-
C...  TION OF THE PRECEDING TAYLOR SERIES, TRUNCATED AFTER THE U
C...                                                          4X
C...  TERMS, FOR U1  GIVES THE FOLLOWING FIVE-POINT APPROXIMATION
C...               X
C...                                                            4
C...  UN  = (1/12DX)(3UN-4 - 16UN-3 + 36UN-2 - 48UN-1 + 25UN) + O(DX )
C...    X
C...                                                           (5)
C...
C...  THE WEIGHTING COEFFICIENTS FOR EQUATIONS (1) TO (5) CAN BE
C...  SUMMARIZED AS
C...
C...              -25   48  -36   16   -3
C...
C...               -3  -10   18   -6    1
C...
C...       1/12     1   -8    0    8   -1
C...
C...               -1    6  -18   10    3
C...
C...                3  -16   36  -48   25
C...
C...  WHICH ARE THE COEFFICIENTS REPORTED BY BICKLEY FOR N = 4, M =
C...  1, P = 0, 1, 2, 3, 4 (BICKLEY, W. G., FORMULAE FOR NUMERICAL
C...  DIFFERENTIATION, MATH. GAZ., VOL. 25, 1941.  NOTE - THE BICKLEY
C...  COEFFICIENTS HAVE BEEN DIVIDED BY A COMMON FACTOR OF TWO).
C...
C...  EQUATIONS (1) TO (5) CAN NOW BE PROGRAMMED TO GENERATE THE
C...  DERIVATIVE U (X) OF FUNCTION U(X) (ARGUMENTS U AND UX OF SUB-
C...              X
C...  ROUTINE DSS004 RESPECTIVELY).
C...
      DIMENSION U(N),UX(N)
C...
C...  COMPUTE THE SPATIAL INCREMENT
      DX=(XU-XL)/FLOAT(N-1)
      R4FDX=1./(12.*DX)
      NM2=N-2
C...
C...  EQUATION (1) (NOTE - THE RHS OF EQUATIONS (1), (2), (3), (4)
C...  AND (5) HAVE BEEN FORMATTED SO THAT THE NUMERICAL WEIGHTING
C...  COEFFICIENTS CAN BE MORE EASILY ASSOCIATED WITH THE BICKLEY
C...  MATRIX ABOVE)
      UX(  1)=R4FDX*
```

```
     1( -25.*U(  1) +48.*U(  2) -36.*U(  3) +16.*U(  4)  -3.*U(  5))
C...
C...  EQUATION (2)
      UX(  2)=R4FDX*
     1(  -3.*U(  1) -10.*U(  2) +18.*U(  3)  -6.*U(  4)  +1.*U(  5))
C...
C...  EQUATION (3)
      DO 1 I=3,NM2
      UX(  I)=R4FDX*
     1(  +1.*U(I-2)  -8.*U(I-1)  +0.*U(  I)  +8.*U(I+1)  -1.*U(I+2))
1     CONTINUE
C...
C...  EQUATION (4)
      UX(N-1)=R4FDX*
     1(  -1.*U(N-4)  +6.*U(N-3) -18.*U(N-2) +10.*U(N-1)  +3.*U(  N))
C...
C...  EQUATION (5)
      UX(  N)=R4FDX*
     1(   3.*U(N-4) -16.*U(N-3) +36.*U(N-2) -48.*U(N-1) +25.*U(  N))
      RETURN
      END
C
      SUBROUTINE DSS020(XL,XU,N,U,UX,V)
C...
C...  SUBROUTINE DSS020 IS AN APPLICATION OF FOURTH-ORDER DIRECTIONAL
C...  DIFFERENCING IN THE NUMERICAL METHOD OF LINES.  IT IS INTENDED
C...  SPECIFICALLY FOR THE ANALYSIS OF CONVECTIVE SYSTEMS MODELLED BY
C...  FIRST-ORDER HYPERBOLIC PARTIAL DIFFERENTIAL EQUATIONS AS DIS-
C...  CUSSED IN SUBROUTINE DSS012.  THE COEFFICIENTS OF THE FINITE
C...  DIFFERENCE APPROXIMATIONS USED HEREIN ARE TAKEN FROM BICKLEY, W.
C...  G., FORMULAE FOR NUMERICAL DIFFERENTIATION, THE MATHEMATICAL
C...  GAZETTE, PP. 19-27, 1941, N = 4, M = 1, P = 0, 1, 2, 3, 4.  THE
C...  IMPLEMENTATION IS THE **FIVE-POINT BIASED UPWIND FORMULA** OF
C...  M. B. CARVER AND H. W. HINDS, THE METHOD OF LINES AND THE
C...  ADVECTION EQUATION, SIMULATION, VOL. 31, NO. 2, PP. 59-69,
C...  AUGUST, 1978
C...
      DIMENSION U(N),UX(N)
C...
C...  COMPUTE THE COMMON FACTOR FOR EACH FINITE DIFFERENCE APPROXIMATION
C...  CONTAINING THE SPATIAL INCREMENT, THEN SELECT THE FINITE DIFFER-
C...  ENCE APPROXIMATION DEPENDING ON THE SIGN OF V (SIXTH ARGUMENT).
      DX=(XU-XL)/FLOAT(N-1)
      R4FDX=1./(12.*DX)
      IF(V.LT.0.)GO TO 10
C...
C...     (1)  FINITE DIFFERENCE APPROXIMATION FOR POSITIVE V
      UX(  1)=R4FDX*
     1( -25.*U(  1) +48.*U(  2) -36.*U(  3) +16.*U(  4)  -3.*U(  5))
      UX(  2)=R4FDX*
     1(  -3.*U(  1) -10.*U(  2) +18.*U(  3)  -6.*U(  4)  +1.*U(  5))
      UX(  3)=R4FDX*
     1(  +1.*U(  1)  -8.*U(  2)  +0.*U(  3)  +8.*U(  4)  -1.*U(  5))
      NM1=N-1
      DO 1 I=4,NM1
      UX(  I)=R4FDX*
     1(  -1.*U(I-3)  +6.*U(I-2) -18.*U(I-1) +10.*U(  I)  +3.*U(I+1))
1     CONTINUE
      UX(  N)=R4FDX*
     1(   3.*U(N-4) -16.*U(N-3) +36.*U(N-2) -48.*U(N-1) +25.*U(  N))
```

```
      RETURN
C...
C...      (2)  FINITE DIFFERENCE APPROXIMATION FOR NEGATIVE V
 10   UX(  1)=R4FDX*
     1( -25.*U(  1) +48.*U(  2) -36.*U(  3) +16.*U(  4)  -3.*U(  5))
      NM3=N-3
      DO 2 I=2,NM3
      UX(  I)=R4FDX*
     1(  -3.*U(I-1) -10.*U(  I) +18.*U(I+1)  -6.*U(I+2)  +1.*U(I+3))
    2 CONTINUE
      UX(N-2)=R4FDX*
     1(  +1.*U(N-4)  -8.*U(N-3)  +0.*U(N-2)  +8.*U(N-1)  -1.*U(  N))
      UX(N-1)=R4FDX*
     1(  -1.*U(N-4)  +6.*U(N-3) -18.*U(N-2) +10.*U(N-1)  +3.*U(  N))
      UX(  N)=R4FDX*
     1(   3.*U(N-4) -16.*U(N-3) +36.*U(N-2) -48.*U(N-1) +25.*U(  N))
      RETURN
      END
```

```
      SUBROUTINE LSODES (F, NEQ, Y, T, TOUT, ITOL, RTOL, ATOL, ITASK,
     1   .         ISTATE, IOPT, RWORK, LRW, IWORK, LIW, JAC, MF)
      EXTERNAL F, JAC
      INTEGER NEQ, ITOL, ITASK, ISTATE, IOPT, LRW, IWORK, LIW, MF
      REAL Y, T, TOUT, RTOL, ATOL, RWORK
      DIMENSION NEQ(1), Y(1), RTOL(1), ATOL(1), RWORK(LRW), IWORK(LIW)
C-----------------------------------------------------------------------
C THIS IS THE MAY 2, 1983 VERSION OF
C LSODES.. LIVERMORE SOLVER FOR ORDINARY DIFFERENTIAL EQUATIONS
C          WITH GENERAL SPARSE JACOBIAN MATRICES.
C THIS VERSION IS IN SINGLE PRECISION.
C
C LSODES SOLVES THE INITIAL VALUE PROBLEM FOR STIFF OR NONSTIFF
C SYSTEMS OF FIRST ORDER ODE-S,
C     DY/DT = F(T,Y) ,  OR, IN COMPONENT FORM,
C     DY(I)/DT = F(I) = F(I,T,Y(1),Y(2),...,Y(NEQ)) (I = 1,...,NEQ).
C LSODES IS A VARIANT OF THE LSODE PACKAGE, AND IS INTENDED FOR
C PROBLEMS IN WHICH THE JACOBIAN MATRIX DF/DY HAS AN ARBITRARY
C SPARSE STRUCTURE (WHEN THE PROBLEM IS STIFF).
C
C AUTHORS..       ALAN C. HINDMARSH,
C                 MATHEMATICS AND STATISTICS DIVISION, L-316
C                 LAWRENCE LIVERMORE NATIONAL LABORATORY
C                 LIVERMORE, CA 94550.
C
C AND             ANDREW H. SHERMAN
C                 EXXON PRODUCTION RESEARCH CO.
C                 P. O. BOX 2189
C                 HOUSTON, TX 77001
C-----------------------------------------------------------------------
C REFERENCES..
C 1.  ALAN C. HINDMARSH,  LSODE AND LSODI, TWO NEW INITIAL VALUE
C     ORDINARY DIFFERENTIAL EQUATION SOLVERS,
C     ACM-SIGNUM NEWSLETTER, VOL. 15, NO. 4 (1980), PP. 10-11.
C
C 2.  S. C. EISENSTAT, M. C. GURSKY, M. H. SCHULTZ, AND A. H. SHERMAN,
C     YALE SPARSE MATRIX PACKAGE.. I. THE SYMMETRIC CODES,
C     RESEARCH REPORT NO. 112, DEPT. OF COMPUTER SCIENCES, YALE
C     UNIVERSITY, 1977.
C
C 3.  S. C. EISENSTAT, M. C. GURSKY, M. H. SCHULTZ, AND A. H. SHERMAN,
C     YALE SPARSE MATRIX PACKAGE.. II. THE NONSYMMETRIC CODES,
C     RESEARCH REPORT NO. 114, DEPT. OF COMPUTER SCIENCES, YALE
C     UNIVERSITY, 1977.
C-----------------------------------------------------------------------
C SUMMARY OF USAGE.
C
C COMMUNICATION BETWEEN THE USER AND THE LSODES PACKAGE, FOR NORMAL
C SITUATIONS, IS SUMMARIZED HERE.  THIS SUMMARY DESCRIBES ONLY A SUBSET
C OF THE FULL SET OF OPTIONS AVAILABLE.  SEE THE FULL DESCRIPTION FOR
C DETAILS, INCLUDING OPTIONAL COMMUNICATION, NONSTANDARD OPTIONS,
C AND INSTRUCTIONS FOR SPECIAL SITUATIONS.  SEE ALSO THE EXAMPLE
C PROBLEM (WITH PROGRAM AND OUTPUT) FOLLOWING THIS SUMMARY.
C
C A. FIRST PROVIDE A SUBROUTINE OF THE FORM..
C                 SUBROUTINE F (NEQ, T, Y, YDOT)
C                 DIMENSION Y(NEQ), YDOT(NEQ)
C WHICH SUPPLIES THE VECTOR FUNCTION F BY LOADING YDOT(I) WITH F(I).
C
C B. NEXT DETERMINE (OR GUESS) WHETHER OR NOT THE PROBLEM IS STIFF.
```

```
      STIFFNESS OCCURS WHEN THE JACOBIAN MATRIX DF/DY HAS AN EIGENVALUE
C     WHOSE REAL PART IS NEGATIVE AND LARGE IN MAGNITUDE, COMPARED TO THE
      RECIPROCAL OF THE T SPAN OF INTEREST.  IF THE PROBLEM IS NONSTIFF,
      USE A METHOD FLAG MF = 10.  IF IT IS STIFF, THERE ARE TWO STANDARD
C     FOR THE METHOD FLAG, MF = 121 AND MF = 222.  IN BOTH CASES, LSODES
C     REQUIRES THE JACOBIAN MATRIX IN SOME FORM, AND IT TREATS THIS MATRIX
      IN GENERAL SPARSE FORM, WITH SPARSITY STRUCTURE DETERMINED INTERNALLY.
      (FOR OPTIONS WHERE THE USER SUPPLIES THE SPARSITY STRUCTURE, SEE
C     THE FULL DESCRIPTION OF MF BELOW.)

      C. IF THE PROBLEM IS STIFF, YOU ARE ENCOURAGED TO SUPPLY THE JACOBIAN
C     DIRECTLY (MF = 121), BUT IF THIS IS NOT FEASIBLE, LSODES WILL
C     COMPUTE IT INTERNALLY BY DIFFERENCE QUOTIENTS (MF = 222).
      IF YOU ARE SUPPLYING THE JACOBIAN, PROVIDE A SUBROUTINE OF THE FORM..
                  SUBROUTINE JAC (NEQ, T, Y, J, IAN, JAN, PDJ)
C                 DIMENSION Y(1), IAN(1), JAN(1), PDJ(1)
      HERE NEQ, T, Y, AND J ARE INPUT ARGUMENTS, AND THE JAC ROUTINE IS TO
      LOAD THE ARRAY PDJ (OF LENGTH NEQ) WITH THE J-TH COLUMN OF DF/DY.
C     I.E., LOAD PDJ(I) WITH DF(I)/DY(J) FOR ALL RELEVANT VALUES OF I.
C     THE ARGUMENTS IAN AND JAN SHOULD BE IGNORED FOR NORMAL SITUATIONS.
      LSODES WILL CALL THE JAC ROUTINE WITH J = 1,2,...,NEQ.
      ONLY NONZERO ELEMENTS NEED BE LOADED.  USUALLY, A CRUDE APPROXIMATION
C     TO DF/DY, POSSIBLY WITH FEWER NONZERO ELEMENTS, WILL SUFFICE.

      D. WRITE A MAIN PROGRAM WHICH CALLS SUBROUTINE LSODES ONCE FOR
C     EACH POINT AT WHICH ANSWERS ARE DESIRED.  THIS SHOULD ALSO PROVIDE
C     FOR POSSIBLE USE OF LOGICAL UNIT 6 FOR OUTPUT OF ERROR MESSAGES
      BY LSODES.  ON THE FIRST CALL TO LSODES, SUPPLY ARGUMENTS AS FOLLOWS..
C     F      = NAME OF SUBROUTINE FOR RIGHT-HAND SIDE VECTOR F.
C              THIS NAME MUST BE DECLARED EXTERNAL IN CALLING PROGRAM.
      NEQ    = NUMBER OF FIRST ORDER ODE-S.
      Y      = ARRAY OF INITIAL VALUES, OF LENGTH NEQ.
C     T      = THE INITIAL VALUE OF THE INDEPENDENT VARIABLE.
      TOUT   = FIRST POINT WHERE OUTPUT IS DESIRED (.NE. T).
      ITOL   = 1 OR 2 ACCORDING AS ATOL (BELOW) IS A SCALAR OR ARRAY.
C     RTOL   = RELATIVE TOLERANCE PARAMETER (SCALAR).
C     ATOL   = ABSOLUTE TOLERANCE PARAMETER (SCALAR OR ARRAY).
               THE ESTIMATED LOCAL ERROR IN Y(I) WILL BE CONTROLLED SO AS
               TO BE ROUGHLY LESS (IN MAGNITUDE) THAN
C                 EWT(I) = RTOL*ABS(Y(I)) + ATOL     IF ITOL = 1, OR
                  EWT(I) = RTOL*ABS(Y(I)) + ATOL(I)  IF ITOL = 2.
               THUS THE LOCAL ERROR TEST PASSES IF, IN EACH COMPONENT,
C              EITHER THE ABSOLUTE ERROR IS LESS THAN ATOL (OR ATOL(I)),
               OR THE RELATIVE ERROR IS LESS THAN RTOL.
               USE RTOL = 0.0 FOR PURE ABSOLUTE ERROR CONTROL, AND
C              USE ATOL = 0.0 (OR ATOL(I) = 0.0) FOR PURE RELATIVE ERROR
C              CONTROL.  CAUTION.. ACTUAL (GLOBAL) ERRORS MAY EXCEED THESE
               LOCAL TOLERANCES, SO CHOOSE THEM CONSERVATIVELY.
      ITASK  = 1 FOR NORMAL COMPUTATION OF OUTPUT VALUES OF Y AT T = TOUT.
C     ISTATE = INTEGER FLAG (INPUT AND OUTPUT).  SET ISTATE = 1.
      IOPT   = 0 TO INDICATE NO OPTIONAL INPUTS USED.
      RWORK  = REAL WORK ARRAY OF LENGTH AT LEAST..
C                 20 + 16*NEQ                    FOR MF = 10,
                  20 + (2 + 1./LENRAT)*NNZ + (11 + 9./LENRAT)*NEQ
                                                 FOR MF = 121 OR 222,
               WHERE..
C              NNZ    = THE NUMBER OF NONZERO ELEMENTS IN THE SPARSE
                        JACOBIAN (IF THIS IS UNKNOWN, USE AN ESTIMATE), AND
               LENRAT = THE REAL TO INTEGER WORDLENGTH RATIO (USUALLY 1 IN
C                       SINGLE PRECISION AND 2 IN DOUBLE PRECISION).
```

```
C          IN ANY CASE, THE REQUIRED SIZE OF RWORK CANNOT GENERALLY
C          BE PREDICTED IN ADVANCE IF MF = 121 OR 222, AND THE VALUE
C          ABOVE IS A ROUGH ESTIMATE OF A CRUDE LOWER BOUND.  SOME
C          EXPERIMENTATION WITH THIS SIZE MAY BE NECESSARY.
C          (WHEN KNOWN, THE CORRECT REQUIRED LENGTH IS AN OPTIONAL
C          OUTPUT, AVAILABLE IN IWORK(17).)
C LRW     = DECLARED LENGTH OF RWORK (IN USER-S DIMENSION).
C IWORK   = INTEGER WORK ARRAY OF LENGTH AT LEAST 30.
C LIW     = DECLARED LENGTH OF IWORK (IN USER-S DIMENSION).
C JAC     = NAME OF SUBROUTINE FOR JACOBIAN MATRIX (MF = 121).
C          IF USED, THIS NAME MUST BE DECLARED EXTERNAL IN CALLING
C          PROGRAM.  IF NOT USED, PASS A DUMMY NAME.
C MF      = METHOD FLAG.  STANDARD VALUES ARE..
C          10  FOR NONSTIFF (ADAMS) METHOD, NO JACOBIAN USED.
C          121 FOR STIFF (BDF) METHOD, USER-SUPPLIED SPARSE JACOBIAN.
C          222 FOR STIFF METHOD, INTERNALLY GENERATED SPARSE JACOBIAN.
C NOTE THAT THE MAIN PROGRAM MUST DECLARE ARRAYS Y, RWORK, IWORK,
C AND POSSIBLY ATOL.
C
C E. THE OUTPUT FROM THE FIRST CALL (OR ANY CALL) IS..
C      Y = ARRAY OF COMPUTED VALUES OF Y(T) VECTOR.
C      T = CORRESPONDING VALUE OF INDEPENDENT VARIABLE (NORMALLY TOUT).
C ISTATE = 2  IF LSODES WAS SUCCESSFUL, NEGATIVE OTHERWISE.
C          -1 MEANS EXCESS WORK DONE ON THIS CALL (PERHAPS WRONG MF).
C          -2 MEANS EXCESS ACCURACY REQUESTED (TOLERANCES TOO SMALL).
C          -3 MEANS ILLEGAL INPUT DETECTED (SEE PRINTED MESSAGE).
C          -4 MEANS REPEATED ERROR TEST FAILURES (CHECK ALL INPUTS).
C          -5 MEANS REPEATED CONVERGENCE FAILURES (PERHAPS BAD JACOBIAN
C             SUPPLIED OR WRONG CHOICE OF MF OR TOLERANCES).
C          -6 MEANS ERROR WEIGHT BECAME ZERO DURING PROBLEM. (SOLUTION
C             COMPONENT I VANISHED, AND ATOL OR ATOL(I) = 0.)
C          A RETURN WITH ISTATE = -1, -4, OR -5 MAY RESULT FROM USING
C          AN INAPPROPRIATE SPARSITY STRUCTURE, ONE THAT IS QUITE
C          DIFFERENT FROM THE INITIAL STRUCTURE.  CONSIDER CALLING
C          LSODES AGAIN WITH ISTATE = 3 TO FORCE THE STRUCTURE TO BE
C          REEVALUATED.  SEE THE FULL DESCRIPTION OF ISTATE BELOW.
C
C F. TO CONTINUE THE INTEGRATION AFTER A SUCCESSFUL RETURN, SIMPLY
C RESET TOUT AND CALL LSODES AGAIN.  NO OTHER PARAMETERS NEED BE RESET.
C
C-----------------------------------------------------------------------
C FULL DESCRIPTION OF USER INTERFACE TO LSODES.
C
C THE USER INTERFACE TO LSODES CONSISTS OF THE FOLLOWING PARTS.
C
C I.    THE CALL SEQUENCE TO SUBROUTINE LSODES, WHICH IS A DRIVER
C       ROUTINE FOR THE SOLVER.  THIS INCLUDES DESCRIPTIONS OF BOTH
C       THE CALL SEQUENCE ARGUMENTS AND OF USER-SUPPLIED ROUTINES.
C       FOLLOWING THESE DESCRIPTIONS IS A DESCRIPTION OF
C       OPTIONAL INPUTS AVAILABLE THROUGH THE CALL SEQUENCE, AND THEN
C       A DESCRIPTION OF OPTIONAL OUTPUTS (IN THE WORK ARRAYS).
C
C II.   DESCRIPTIONS OF OTHER ROUTINES IN THE LSODES PACKAGE THAT MAY BE
C       (OPTIONALLY) CALLED BY THE USER.  THESE PROVIDE THE ABILITY TO
C       ALTER ERROR MESSAGE HANDLING, SAVE AND RESTORE THE INTERNAL
C       COMMON, AND OBTAIN SPECIFIED DERIVATIVES OF THE SOLUTION Y(T).
C
C III.  DESCRIPTIONS OF COMMON BLOCKS TO BE DECLARED IN OVERLAY
C       OR SIMILAR ENVIRONMENTS, OR TO BE SAVED WHEN DOING AN INTERRUPT
C       OF THE PROBLEM AND CONTINUED SOLUTION LATER.
```

```
C IV.  DESCRIPTION OF TWO SUBROUTINES IN THE LSODES PACKAGE, EITHER OF
       WHICH THE USER MAY REPLACE WITH HIS OWN VERSION, IF DESIRED.
       THESE RELATE TO THE MEASUREMENT OF ERRORS.
C
C-----------------------------------------------------------------------
  PART I.  CALL SEQUENCE.

C THE CALL SEQUENCE PARAMETERS USED FOR INPUT ONLY ARE
      F, NEQ, TOUT, ITOL, RTOL, ATOL, ITASK, IOPT, LRW, LIW, JAC, MF,
  AND THOSE USED FOR BOTH INPUT AND OUTPUT ARE
C     Y, T, ISTATE.
C THE WORK ARRAYS RWORK AND IWORK ARE ALSO USED FOR CONDITIONAL AND
  OPTIONAL INPUTS AND OPTIONAL OUTPUTS.  (THE TERM OUTPUT HERE REFERS
C TO THE RETURN FROM SUBROUTINE LSODES TO THE USER-S CALLING PROGRAM.)
C
  THE LEGALITY OF INPUT PARAMETERS WILL BE THOROUGHLY CHECKED ON THE
  INITIAL CALL FOR THE PROBLEM, BUT NOT CHECKED THEREAFTER UNLESS A
C CHANGE IN INPUT PARAMETERS IS FLAGGED BY ISTATE = 3 ON INPUT.

  THE DESCRIPTIONS OF THE CALL ARGUMENTS ARE AS FOLLOWS.
C
C F       = THE NAME OF THE USER-SUPPLIED SUBROUTINE DEFINING THE
            ODE SYSTEM.  THE SYSTEM MUST BE PUT IN THE FIRST-ORDER
            FORM DY/DT = F(T,Y), WHERE F IS A VECTOR-VALUED FUNCTION
C           OF THE SCALAR T AND THE VECTOR Y.  SUBROUTINE F IS TO
            COMPUTE THE FUNCTION F.  IT IS TO HAVE THE FORM
                 SUBROUTINE F (NEQ, T, Y, YDOT)
C                DIMENSION Y(1), YDOT(1)
C           WHERE NEQ, T, AND Y ARE INPUT, AND THE ARRAY YDOT = F(T,Y)
            IS OUTPUT.  Y AND YDOT ARE ARRAYS OF LENGTH NEQ.
            (IN THE DIMENSION STATEMENT ABOVE, 1 IS A DUMMY
C           DIMENSION.. IT CAN BE REPLACED BY ANY VALUE.)
            SUBROUTINE F SHOULD NOT ALTER Y(1),...,Y(NEQ).
            F MUST BE DECLARED EXTERNAL IN THE CALLING PROGRAM.
C
            SUBROUTINE F MAY ACCESS USER-DEFINED QUANTITIES IN
            NEQ(2),... AND Y(NEQ(1)+1),... IF NEQ IS AN ARRAY
C           (DIMENSIONED IN F) AND Y HAS LENGTH EXCEEDING NEQ(1).
C           SEE THE DESCRIPTIONS OF NEQ AND Y BELOW.

  NEQ     = THE SIZE OF THE ODE SYSTEM (NUMBER OF FIRST ORDER
C           ORDINARY DIFFERENTIAL EQUATIONS).  USED ONLY FOR INPUT.
            NEQ MAY BE DECREASED, BUT NOT INCREASED, DURING THE PROBLEM.
            IF NEQ IS DECREASED (WITH ISTATE = 3 ON INPUT), THE
C           REMAINING COMPONENTS OF Y SHOULD BE LEFT UNDISTURBED, IF
            THESE ARE TO BE ACCESSED IN F AND/OR JAC.

C           NORMALLY, NEQ IS A SCALAR, AND IT IS GENERALLY REFERRED TO
C           AS A SCALAR IN THIS USER INTERFACE DESCRIPTION.  HOWEVER,
            NEQ MAY BE AN ARRAY, WITH NEQ(1) SET TO THE SYSTEM SIZE.
            (THE LSODES PACKAGE ACCESSES ONLY NEQ(1).)  IN EITHER CASE,
C           THIS PARAMETER IS PASSED AS THE NEQ ARGUMENT IN ALL CALLS
            TO F AND JAC.  HENCE, IF IT IS AN ARRAY, LOCATIONS
            NEQ(2),... MAY BE USED TO STORE OTHER INTEGER DATA AND PASS
C           IT TO F AND/OR JAC.  SUBROUTINES F AND/OR JAC MUST INCLUDE
C           NEQ IN A DIMENSION STATEMENT IN THAT CASE.

C Y       = A REAL ARRAY FOR THE VECTOR OF DEPENDENT VARIABLES, OF
C           LENGTH NEQ OR MORE.  USED FOR BOTH INPUT AND OUTPUT ON THE
```

```
C          FIRST CALL (ISTATE = 1), AND ONLY FOR OUTPUT ON OTHER CALLS.
C          ON THE FIRST CALL, Y MUST CONTAIN THE VECTOR OF INITIAL
C          VALUES.  ON OUTPUT, Y CONTAINS THE COMPUTED SOLUTION VECTOR,
C          EVALUATED AT T.  IF DESIRED, THE Y ARRAY MAY BE USED
C          FOR OTHER PURPOSES BETWEEN CALLS TO THE SOLVER.
C
C          THIS ARRAY IS PASSED AS THE Y ARGUMENT IN ALL CALLS TO
C          F AND JAC.  HENCE ITS LENGTH MAY EXCEED NEQ, AND LOCATIONS
C          Y(NEQ+1),... MAY BE USED TO STORE OTHER REAL DATA AND
C          PASS IT TO F AND/OR JAC.  (THE LSODES PACKAGE ACCESSES ONLY
C          Y(1),...,Y(NEQ).)
C
C T      = THE INDEPENDENT VARIABLE.  ON INPUT, T IS USED ONLY ON THE
C          FIRST CALL, AS THE INITIAL POINT OF THE INTEGRATION.
C          ON OUTPUT, AFTER EACH CALL, T IS THE VALUE AT WHICH A
C          COMPUTED SOLUTION Y IS EVALUATED (USUALLY THE SAME AS TOUT).
C          ON AN ERROR RETURN, T IS THE FARTHEST POINT REACHED.
C
C TOUT   = THE NEXT VALUE OF T AT WHICH A COMPUTED SOLUTION IS DESIRED.
C          USED ONLY FOR INPUT.
C
C          WHEN STARTING THE PROBLEM (ISTATE = 1), TOUT MAY BE EQUAL
C          TO T FOR ONE CALL, THEN SHOULD .NE. T FOR THE NEXT CALL.
C          FOR THE INITIAL T, AN INPUT VALUE OF TOUT .NE. T IS USED
C          IN ORDER TO DETERMINE THE DIRECTION OF THE INTEGRATION
C          (I.E. THE ALGEBRAIC SIGN OF THE STEP SIZES) AND THE ROUGH
C          SCALE OF THE PROBLEM.  INTEGRATION IN EITHER DIRECTION
C          (FORWARD OR BACKWARD IN T) IS PERMITTED.
C
C          IF ITASK = 2 OR 5 (ONE-STEP MODES), TOUT IS IGNORED AFTER
C          THE FIRST CALL (I.E. THE FIRST CALL WITH TOUT .NE. T).
C          OTHERWISE, TOUT IS REQUIRED ON EVERY CALL.
C
C          IF ITASK = 1, 3, OR 4, THE VALUES OF TOUT NEED NOT BE
C          MONOTONE, BUT A VALUE OF TOUT WHICH BACKS UP IS LIMITED
C          TO THE CURRENT INTERNAL T INTERVAL, WHOSE ENDPOINTS ARE
C          TCUR - HU AND TCUR (SEE OPTIONAL OUTPUTS, BELOW, FOR
C          TCUR AND HU).
C
C ITOL   = AN INDICATOR FOR THE TYPE OF ERROR CONTROL.  SEE
C          DESCRIPTION BELOW UNDER ATOL.  USED ONLY FOR INPUT.
C
C RTOL   = A RELATIVE ERROR TOLERANCE PARAMETER, EITHER A SCALAR OR
C          AN ARRAY OF LENGTH NEQ.  SEE DESCRIPTION BELOW UNDER ATOL.
C          INPUT ONLY.
C
C ATOL   = AN ABSOLUTE ERROR TOLERANCE PARAMETER, EITHER A SCALAR OR
C          AN ARRAY OF LENGTH NEQ.  INPUT ONLY.
C
C             THE INPUT PARAMETERS ITOL, RTOL, AND ATOL DETERMINE
C          THE ERROR CONTROL PERFORMED BY THE SOLVER.  THE SOLVER WILL
C          CONTROL THE VECTOR E = (E(I)) OF ESTIMATED LOCAL ERRORS
C          IN Y, ACCORDING TO AN INEQUALITY OF THE FORM
C                      RMS-NORM OF ( E(I)/EWT(I) )   .LE.   1,
C          WHERE       EWT(I) = RTOL(I)*ABS(Y(I)) + ATOL(I),
C          AND THE RMS-NORM (ROOT-MEAN-SQUARE NORM) HERE IS
C          RMS-NORM(V) = SQRT(SUM V(I)**2 / NEQ).  HERE EWT = (EWT(I))
C          IS A VECTOR OF WEIGHTS WHICH MUST ALWAYS BE POSITIVE, AND
C          THE VALUES OF RTOL AND ATOL SHOULD ALL BE NON-NEGATIVE.
C          THE FOLLOWING TABLE GIVES THE TYPES (SCALAR/ARRAY) OF
```

```
C       RTOL AND ATOL, AND THE CORRESPONDING FORM OF EWT(I).
C
C          ITOL    RTOL       ATOL          EWT(I)
C           1      SCALAR     SCALAR     RTOL*ABS(Y(I)) + ATOL
C           2      SCALAR     ARRAY      RTOL*ABS(Y(I)) + ATOL(I)
C           3      ARRAY      SCALAR     RTOL(I)*ABS(Y(I)) + ATOL
C           4      ARRAY      ARRAY      RTOL(I)*ABS(Y(I)) + ATOL(I)
C
C       WHEN EITHER OF THESE PARAMETERS IS A SCALAR, IT NEED NOT
C       BE DIMENSIONED IN THE USER-S CALLING PROGRAM.
C
C       IF NONE OF THE ABOVE CHOICES (WITH ITOL, RTOL, AND ATOL
C       FIXED THROUGHOUT THE PROBLEM) IS SUITABLE, MORE GENERAL
C       ERROR CONTROLS CAN BE OBTAINED BY SUBSTITUTING
C       USER-SUPPLIED ROUTINES FOR THE SETTING OF EWT AND/OR FOR
C       THE NORM CALCULATION.  SEE PART IV BELOW.
C
C       IF GLOBAL ERRORS ARE TO BE ESTIMATED BY MAKING A REPEATED
C       RUN ON THE SAME PROBLEM WITH SMALLER TOLERANCES, THEN ALL
C       COMPONENTS OF RTOL AND ATOL (I.E. OF EWT) SHOULD BE SCALED
C       DOWN UNIFORMLY.
C
C ITASK  = AN INDEX SPECIFYING THE TASK TO BE PERFORMED.
C          INPUT ONLY.  ITASK HAS THE FOLLOWING VALUES AND MEANINGS.
C          1  MEANS NORMAL COMPUTATION OF OUTPUT VALUES OF Y(T) AT
C             T = TOUT (BY OVERSHOOTING AND INTERPOLATING).
C          2  MEANS TAKE ONE STEP ONLY AND RETURN.
C          3  MEANS STOP AT THE FIRST INTERNAL MESH POINT AT OR
C             BEYOND T = TOUT AND RETURN.
C          4  MEANS NORMAL COMPUTATION OF OUTPUT VALUES OF Y(T) AT
C             T = TOUT BUT WITHOUT OVERSHOOTING T = TCRIT.
C             TCRIT MUST BE INPUT AS RWORK(1).  TCRIT MAY BE EQUAL TO
C             OR BEYOND TOUT, BUT NOT BEHIND IT IN THE DIRECTION OF
C             INTEGRATION.  THIS OPTION IS USEFUL IF THE PROBLEM
C             HAS A SINGULARITY AT OR BEYOND T = TCRIT.
C          5  MEANS TAKE ONE STEP, WITHOUT PASSING TCRIT, AND RETURN.
C             TCRIT MUST BE INPUT AS RWORK(1).
C
C          NOTE..  IF ITASK = 4 OR 5 AND THE SOLVER REACHES TCRIT
C          (WITHIN ROUNDOFF), IT WILL RETURN T = TCRIT (EXACTLY) TO
C          INDICATE THIS (UNLESS ITASK = 4 AND TOUT COMES BEFORE TCRIT,
C          IN WHICH CASE ANSWERS AT T = TOUT ARE RETURNED FIRST).
C
C ISTATE = AN INDEX USED FOR INPUT AND OUTPUT TO SPECIFY THE
C          THE STATE OF THE CALCULATION.
C
C          ON INPUT, THE VALUES OF ISTATE ARE AS FOLLOWS.
C          1  MEANS THIS IS THE FIRST CALL FOR THE PROBLEM
C             (INITIALIZATIONS WILL BE DONE).  SEE NOTE BELOW.
C          2  MEANS THIS IS NOT THE FIRST CALL, AND THE CALCULATION
C             IS TO CONTINUE NORMALLY, WITH NO CHANGE IN ANY INPUT
C             PARAMETERS EXCEPT POSSIBLY TOUT AND ITASK.
C             (IF ITOL, RTOL, AND/OR ATOL ARE CHANGED BETWEEN CALLS
C             WITH ISTATE = 2, THE NEW VALUES WILL BE USED BUT NOT
C             TESTED FOR LEGALITY.)
C          3  MEANS THIS IS NOT THE FIRST CALL, AND THE
C             CALCULATION IS TO CONTINUE NORMALLY, BUT WITH
C             A CHANGE IN INPUT PARAMETERS OTHER THAN
C             TOUT AND ITASK.  CHANGES ARE ALLOWED IN
C             NEQ, ITOL, RTOL, ATOL, IOPT, LRW, LIW, MF,
```

```
C          THE CONDITIONAL INPUTS IA AND JA,
C          AND ANY OF THE OPTIONAL INPUTS EXCEPT H0.
C          IN PARTICULAR, IF MITER = 1 OR 2, A CALL WITH ISTATE = 3
C          WILL CAUSE THE SPARSITY STRUCTURE OF THE PROBLEM TO BE
C          RECOMPUTED (OR REREAD FROM IA AND JA IF MOSS = 0).
C        NOTE..  A PRELIMINARY CALL WITH TOUT = T IS NOT COUNTED
C        AS A FIRST CALL HERE, AS NO INITIALIZATION OR CHECKING OF
C        INPUT IS DONE.  (SUCH A CALL IS SOMETIMES USEFUL FOR THE
C        PURPOSE OF OUTPUTTING THE INITIAL CONDITIONS.)
C        THUS THE FIRST CALL FOR WHICH TOUT .NE. T REQUIRES
C        ISTATE = 1 ON INPUT.
C
C        ON OUTPUT, ISTATE HAS THE FOLLOWING VALUES AND MEANINGS.
C         1  MEANS NOTHING WAS DONE, AS TOUT WAS EQUAL TO T WITH
C            ISTATE = 1 ON INPUT.  (HOWEVER, AN INTERNAL COUNTER WAS
C            SET TO DETECT AND PREVENT REPEATED CALLS OF THIS TYPE.)
C         2  MEANS THE INTEGRATION WAS PERFORMED SUCCESSFULLY.
C        -1  MEANS AN EXCESSIVE AMOUNT OF WORK (MORE THAN MXSTEP
C            STEPS) WAS DONE ON THIS CALL, BEFORE COMPLETING THE
C            REQUESTED TASK, BUT THE INTEGRATION WAS OTHERWISE
C            SUCCESSFUL AS FAR AS T.  (MXSTEP IS AN OPTIONAL INPUT
C            AND IS NORMALLY 500.)  TO CONTINUE, THE USER MAY
C            SIMPLY RESET ISTATE TO A VALUE .GT. 1 AND CALL AGAIN
C            (THE EXCESS WORK STEP COUNTER WILL BE RESET TO 0).
C            IN ADDITION, THE USER MAY INCREASE MXSTEP TO AVOID
C            THIS ERROR RETURN (SEE BELOW ON OPTIONAL INPUTS).
C        -2  MEANS TOO MUCH ACCURACY WAS REQUESTED FOR THE PRECISION
C            OF THE MACHINE BEING USED.  THIS WAS DETECTED BEFORE
C            COMPLETING THE REQUESTED TASK, BUT THE INTEGRATION
C            WAS SUCCESSFUL AS FAR AS T.  TO CONTINUE, THE TOLERANCE
C            PARAMETERS MUST BE RESET, AND ISTATE MUST BE SET
C            TO 3.  THE OPTIONAL OUTPUT TOLSF MAY BE USED FOR THIS
C            PURPOSE.  (NOTE.. IF THIS CONDITION IS DETECTED BEFORE
C            TAKING ANY STEPS, THEN AN ILLEGAL INPUT RETURN
C            (ISTATE = -3) OCCURS INSTEAD.)
C        -3  MEANS ILLEGAL INPUT WAS DETECTED, BEFORE TAKING ANY
C            INTEGRATION STEPS.  SEE WRITTEN MESSAGE FOR DETAILS.
C            NOTE..  IF THE SOLVER DETECTS AN INFINITE LOOP OF CALLS
C            TO THE SOLVER WITH ILLEGAL INPUT, IT WILL CAUSE
C            THE RUN TO STOP.
C        -4  MEANS THERE WERE REPEATED ERROR TEST FAILURES ON
C            ONE ATTEMPTED STEP, BEFORE COMPLETING THE REQUESTED
C            TASK, BUT THE INTEGRATION WAS SUCCESSFUL AS FAR AS T.
C            THE PROBLEM MAY HAVE A SINGULARITY, OR THE INPUT
C            MAY BE INAPPROPRIATE.
C        -5  MEANS THERE WERE REPEATED CONVERGENCE TEST FAILURES ON
C            ONE ATTEMPTED STEP, BEFORE COMPLETING THE REQUESTED
C            TASK, BUT THE INTEGRATION WAS SUCCESSFUL AS FAR AS T.
C            THIS MAY BE CAUSED BY AN INACCURATE JACOBIAN MATRIX,
C            IF ONE IS BEING USED.
C        -6  MEANS EWT(I) BECAME ZERO FOR SOME I DURING THE
C            INTEGRATION.  PURE RELATIVE ERROR CONTROL (ATOL(I)=0.0)
C            WAS REQUESTED ON A VARIABLE WHICH HAS NOW VANISHED.
C            THE INTEGRATION WAS SUCCESSFUL AS FAR AS T.
C
C        NOTE.. AN ERROR RETURN WITH ISTATE = -1, -4, OR -5 AND WITH
C        MITER = 1 OR 2 MAY MEAN THAT THE SPARSITY STRUCTURE OF THE
C        PROBLEM HAS CHANGED SIGNIFICANTLY SINCE IT WAS LAST
C        DETERMINED (OR INPUT).  IN THAT CASE, ONE CAN ATTEMPT TO
C        COMPLETE THE INTEGRATION BY SETTING ISTATE = 3 ON THE NEXT
```

```
C         CALL, SO THAT A NEW STRUCTURE DETERMINATION IS DONE.
C
C         NOTE..  SINCE THE NORMAL OUTPUT VALUE OF ISTATE IS 2,
C         IT DOES NOT NEED TO BE RESET FOR NORMAL CONTINUATION.
C         ALSO, SINCE A NEGATIVE INPUT VALUE OF ISTATE WILL BE
C         REGARDED AS ILLEGAL, A NEGATIVE OUTPUT VALUE REQUIRES THE
C         USER TO CHANGE IT, AND POSSIBLY OTHER INPUTS, BEFORE
C         CALLING THE SOLVER AGAIN.
C
C IOPT  = AN INTEGER FLAG TO SPECIFY WHETHER OR NOT ANY OPTIONAL
C         INPUTS ARE BEING USED ON THIS CALL.  INPUT ONLY.
C         THE OPTIONAL INPUTS ARE LISTED SEPARATELY BELOW.
C         IOPT = 0 MEANS NO OPTIONAL INPUTS ARE BEING USED.
C                  DEFAULT VALUES WILL BE USED IN ALL CASES.
C         IOPT = 1 MEANS ONE OR MORE OPTIONAL INPUTS ARE BEING USED.
C
C RWORK  = A WORK ARRAY USED FOR A MIXTURE OF REAL (SINGLE PRECISION)
C         AND INTEGER WORK SPACE.
C         THE LENGTH OF RWORK (IN REAL WORDS) MUST BE AT LEAST
C            20 + NYH*(MAXORD + 1) + 3*NEQ + LWM    WHERE
C         NYH    = THE INITIAL VALUE OF NEQ,
C         MAXORD = 12 (IF METH = 1) OR 5 (IF METH = 2) (UNLESS A
C                  SMALLER VALUE IS GIVEN AS AN OPTIONAL INPUT),
C         LWM = 0                                          IF MITER = 0,
C         LWM = 2*NNZ + 2*NEQ + (NNZ+9*NEQ)/LENRAT         IF MITER = 1,
C         LWM = 2*NNZ + 2*NEQ + (NNZ+10*NEQ)/LENRAT        IF MITER = 2,
C         LWM = NEQ + 2                                    IF MITER = 3.
C         IN THE ABOVE FORMULAS,
C         NNZ    = NUMBER OF NONZERO ELEMENTS IN THE JACOBIAN MATRIX.
C         LENRAT = THE REAL TO INTEGER WORDLENGTH RATIO (USUALLY 1 IN
C                  SINGLE PRECISION AND 2 IN DOUBLE PRECISION).
C         (SEE THE MF DESCRIPTION FOR METH AND MITER.)
C         THUS IF MAXORD HAS ITS DEFAULT VALUE AND NEQ IS CONSTANT,
C         THE MINIMUM LENGTH OF RWORK IS..
C            20 + 16*NEQ          FOR MF = 10,
C            20 + 16*NEQ + LWM    FOR MF = 11, 111, 211, 12, 112, 212,
C            22 + 17*NEQ          FOR MF = 13,
C            20 +  9*NEQ          FOR MF = 20,
C            20 +  9*NEQ + LWM    FOR MF = 21, 121, 221, 22, 122, 222,
C            22 + 10*NEQ          FOR MF = 23.
C         IF MITER = 1 OR 2, THE ABOVE FORMULA FOR LWM IS ONLY A
C         CRUDE LOWER BOUND.  THE REQUIRED LENGTH OF RWORK CANNOT
C         BE READILY PREDICTED IN GENERAL, AS IT DEPENDS ON THE
C         SPARSITY STRUCTURE OF THE PROBLEM.  SOME EXPERIMENTATION
C         MAY BE NECESSARY.
C
C         THE FIRST 20 WORDS OF RWORK ARE RESERVED FOR CONDITIONAL
C         AND OPTIONAL INPUTS AND OPTIONAL OUTPUTS.
C
C         THE FOLLOWING WORD IN RWORK IS A CONDITIONAL INPUT..
C           RWORK(1) = TCRIT = CRITICAL VALUE OF T WHICH THE SOLVER
C                      IS NOT TO OVERSHOOT.  REQUIRED IF ITASK IS
C                      4 OR 5, AND IGNORED OTHERWISE.  (SEE ITASK.)
C
C LRW    = THE LENGTH OF THE ARRAY RWORK, AS DECLARED BY THE USER.
C         (THIS WILL BE CHECKED BY THE SOLVER.)
C
C IWORK  = AN INTEGER WORK ARRAY.  THE LENGTH OF IWORK MUST BE AT LEAST
C            31 + NEQ + NNZ    IF MOSS = 0 AND MITER = 1 OR 2, OR
C            30                OTHERWISE.
```

```
C           (NNZ IS THE NUMBER OF NONZERO ELEMENTS IN DF/DY.)
C
C           IN LSODES, IWORK IS USED ONLY FOR CONDITIONAL AND
C           OPTIONAL INPUTS AND OPTIONAL OUTPUTS.
C
C           THE FOLLOWING TWO BLOCKS OF WORDS IN IWORK ARE CONDITIONAL
C           INPUTS, REQUIRED IF MOSS = 0 AND MITER = 1 OR 2, BUT NOT
C           OTHERWISE (SEE THE DESCRIPTION OF MF FOR MOSS).
C             IWORK(30+J) = IA(J)      (J=1,...,NEQ+1)
C             IWORK(31+NEQ+K) = JA(K)  (K=1,...,NNZ)
C           THE TWO ARRAYS IA AND JA DESCRIBE THE SPARSITY STRUCTURE
C           TO BE ASSUMED FOR THE JACOBIAN MATRIX.  JA CONTAINS THE ROW
C           INDICES WHERE NONZERO ELEMENTS OCCUR, READING IN COLUMNWISE
C           ORDER, AND IA CONTAINS THE STARTING LOCATIONS IN JA OF THE
C           DESCRIPTIONS OF COLUMNS 1,...,NEQ, IN THAT ORDER, WITH
C           IA(1) = 1.  THUS, FOR EACH COLUMN INDEX J = 1,...,NEQ, THE
C           VALUES OF THE ROW INDEX I IN COLUMN J WHERE A NONZERO
C           ELEMENT MAY OCCUR ARE GIVEN BY
C             I = JA(K),  WHERE   IA(J) .LE. K .LT. IA(J+1).
C           IF NNZ IS THE TOTAL NUMBER OF NONZERO LOCATIONS ASSUMED,
C           THEN THE LENGTH OF THE JA ARRAY IS NNZ, AND IA(NEQ+1) MUST
C           BE NNZ + 1.  DUPLICATE ENTRIES ARE NOT ALLOWED.
C
C LIW     = THE LENGTH OF THE ARRAY IWORK, AS DECLARED BY THE USER.
C           (THIS WILL BE CHECKED BY THE SOLVER.)
C
C NOTE..  THE WORK ARRAYS MUST NOT BE ALTERED BETWEEN CALLS TO LSODES
C FOR THE SAME PROBLEM, EXCEPT POSSIBLY FOR THE CONDITIONAL AND
C OPTIONAL INPUTS, AND EXCEPT FOR THE LAST 3*NEQ WORDS OF RWORK.
C THE LATTER SPACE IS USED FOR INTERNAL SCRATCH SPACE, AND SO IS
C AVAILABLE FOR USE BY THE USER OUTSIDE LSODES BETWEEN CALLS, IF
C DESIRED (BUT NOT FOR USE BY F OR JAC).
C
C JAC     = NAME OF USER-SUPPLIED ROUTINE (MITER = 1 OR MOSS = 1) TO
C           COMPUTE THE JACOBIAN MATRIX, DF/DY, AS A FUNCTION OF
C           THE SCALAR T AND THE VECTOR Y.  IT IS TO HAVE THE FORM
C                SUBROUTINE JAC (NEQ, T, Y, J, IAN, JAN, PDJ)
C                DIMENSION Y(1), IAN(1), JAN(1), PDJ(1)
C           WHERE NEQ, T, Y, J, IAN, AND JAN ARE INPUT, AND THE ARRAY
C           PDJ, OF LENGTH NEQ, IS TO BE LOADED WITH COLUMN J
C           OF THE JACOBIAN ON OUTPUT.  THUS DF(I)/DY(J) IS TO BE
C           LOADED INTO PDJ(I) FOR ALL RELEVANT VALUES OF I.
C           HERE T AND Y HAVE THE SAME MEANING AS IN SUBROUTINE F,
C           AND J IS A COLUMN INDEX (1 TO NEQ).  IAN AND JAN ARE
C           UNDEFINED IN CALLS TO JAC FOR STRUCTURE DETERMINATION
C           (MOSS = 1).  OTHERWISE, IAN AND JAN ARE STRUCTURE
C           DESCRIPTORS, AS DEFINED UNDER OPTIONAL OUTPUTS BELOW, AND
C           SO CAN BE USED TO DETERMINE THE RELEVANT ROW INDICES I, IF
C           DESIRED.  (IN THE DIMENSION STATEMENT ABOVE, 1 IS A
C           DUMMY DIMENSION.. IT CAN BE REPLACED BY ANY VALUE.)
C                JAC NEED NOT PROVIDE DF/DY EXACTLY.  A CRUDE
C           APPROXIMATION (POSSIBLY WITH GREATER SPARSITY) WILL DO.
C                IN ANY CASE, PDJ IS PRESET TO ZERO BY THE SOLVER,
C           SO THAT ONLY THE NONZERO ELEMENTS NEED BE LOADED BY JAC.
C           CALLS TO JAC ARE MADE WITH J = 1,...,NEQ, IN THAT ORDER, AND
C           EACH SUCH SET OF CALLS IS PRECEDED BY A CALL TO F WITH THE
C           SAME ARGUMENTS NEQ, T, AND Y.  THUS TO GAIN SOME EFFICIENCY,
C           INTERMEDIATE QUANTITIES SHARED BY BOTH CALCULATIONS MAY BE
C           SAVED IN A USER COMMON BLOCK BY F AND NOT RECOMPUTED BY JAC,
C           IF DESIRED.  JAC MUST NOT ALTER ITS INPUT ARGUMENTS.
```

```
C          JAC MUST BE DECLARED EXTERNAL IN THE CALLING PROGRAM.
C               SUBROUTINE JAC MAY ACCESS USER-DEFINED QUANTITIES IN
C          NEQ(2),... AND Y(NEQ(1)+1),... IF NEQ IS AN ARRAY
C          (DIMENSIONED IN JAC) AND Y HAS LENGTH EXCEEDING NEQ(1).
C          SEE THE DESCRIPTIONS OF NEQ AND Y ABOVE.
C
C MF     = THE METHOD FLAG.  USED ONLY FOR INPUT.
C          MF HAS THREE DECIMAL DIGITS-- MOSS, METH, MITER--
C             MF = 100*MOSS + 10*METH + MITER.
C          MOSS INDICATES THE METHOD TO BE USED TO OBTAIN THE SPARSITY
C          STRUCTURE OF THE JACOBIAN MATRIX IF MITER = 1 OR 2..
C            MOSS = 0 MEANS THE USER HAS SUPPLIED IA AND JA
C                     (SEE DESCRIPTIONS UNDER IWORK ABOVE).
C            MOSS = 1 MEANS THE USER HAS SUPPLIED JAC (SEE BELOW)
C                     AND THE STRUCTURE WILL BE OBTAINED FROM NEQ
C                     INITIAL CALLS TO JAC.
C            MOSS = 2 MEANS THE STRUCTURE WILL BE OBTAINED FROM NEQ+1
C                     INITIAL CALLS TO F.
C          METH INDICATES THE BASIC LINEAR MULTISTEP METHOD..
C            METH = 1 MEANS THE IMPLICIT ADAMS METHOD.
C            METH = 2 MEANS THE METHOD BASED ON BACKWARD
C                     DIFFERENTIATION FORMULAS (BDF-S).
C          MITER INDICATES THE CORRECTOR ITERATION METHOD..
C            MITER = 0 MEANS FUNCTIONAL ITERATION (NO JACOBIAN MATRIX
C                      IS INVOLVED).
C            MITER = 1 MEANS CHORD ITERATION WITH A USER-SUPPLIED
C                      SPARSE JACOBIAN, GIVEN BY SUBROUTINE JAC.
C            MITER = 2 MEANS CHORD ITERATION WITH AN INTERNALLY
C                      GENERATED (DIFFERENCE QUOTIENT) SPARSE JACOBIAN
C                      (USING NGP EXTRA CALLS TO F PER DF/DY VALUE,
C                      WHERE NGP IS AN OPTIONAL OUTPUT DESCRIBED BELOW.)
C            MITER = 3 MEANS CHORD ITERATION WITH AN INTERNALLY
C                      GENERATED DIAGONAL JACOBIAN APPROXIMATION.
C                      (USING 1 EXTRA CALL TO F PER DF/DY EVALUATION).
C          IF MITER = 1 OR MOSS = 1, THE USER MUST SUPPLY A SUBROUTINE
C          JAC (THE NAME IS ARBITRARY) AS DESCRIBED ABOVE UNDER JAC.
C          OTHERWISE, A DUMMY ARGUMENT CAN BE USED.
C
C          THE STANDARD CHOICES FOR MF ARE..
C            MF = 10  FOR A NONSTIFF PROBLEM,
C            MF = 21 OR 22 FOR A STIFF PROBLEM WITH IA/JA SUPPLIED
C                    (21 IF JAC IS SUPPLIED, 22 IF NOT),
C            MF = 121 FOR A STIFF PROBLEM WITH JAC SUPPLIED,
C                     BUT NOT IA/JA,
C            MF = 222 FOR A STIFF PROBLEM WITH NEITHER IA/JA NOR
C                     JAC SUPPLIED.
C          THE SPARSENESS STRUCTURE CAN BE CHANGED DURING THE
C          PROBLEM BY MAKING A CALL TO LSODES WITH ISTATE = 3.
C-----------------------------------------------------------------------
C OPTIONAL INPUTS.
C
C  THE FOLLOWING IS A LIST OF THE OPTIONAL INPUTS PROVIDED FOR IN THE
C CALL SEQUENCE.  (SEE ALSO PART II.)  FOR EACH SUCH INPUT VARIABLE,
C THIS TABLE LISTS ITS NAME AS USED IN THIS DOCUMENTATION, ITS
C  LOCATION IN THE CALL SEQUENCE, ITS MEANING, AND THE DEFAULT VALUE.
C THE USE OF ANY OF THESE INPUTS REQUIRES IOPT = 1, AND IN THAT
C CASE ALL OF THESE INPUTS ARE EXAMINED.  A VALUE OF ZERO FOR ANY
C  OF THESE OPTIONAL INPUTS WILL CAUSE THE DEFAULT VALUE TO BE USED.
C  THUS TO USE A SUBSET OF THE OPTIONAL INPUTS, SIMPLY PRELOAD
C LOCATIONS 5 TO 10 IN RWORK AND IWORK TO 0.0 AND 0 RESPECTIVELY, AND
```

```
C THEN SET THOSE OF INTEREST TO NONZERO VALUES.
C
C NAME     LOCATION      MEANING AND DEFAULT VALUE
C
C H0       RWORK(5)  THE STEP SIZE TO BE ATTEMPTED ON THE FIRST STEP.
C                    THE DEFAULT VALUE IS DETERMINED BY THE SOLVER.
C
C HMAX     RWORK(6)  THE MAXIMUM ABSOLUTE STEP SIZE ALLOWED.
C                    THE DEFAULT VALUE IS INFINITE.
C
C HMIN     RWORK(7)  THE MINIMUM ABSOLUTE STEP SIZE ALLOWED.
C                    THE DEFAULT VALUE IS 0.  (THIS LOWER BOUND IS NOT
C                    ENFORCED ON THE FINAL STEP BEFORE REACHING TCRIT
C                    WHEN ITASK = 4 OR 5.)
C
C SETH     RWORK(8)  THE ELEMENT THRESHHOLD FOR SPARSITY DETERMINATION
C                    WHEN MOSS = 1 OR 2.  IF THE ABSOLUTE VALUE OF
C                    AN ESTIMATED JACOBIAN ELEMENT IS .LE. SETH, IT
C                    WILL BE ASSUMED TO BE ABSENT IN THE STRUCTURE.
C                    THE DEFAULT VALUE OF SETH IS 0.
C
C MAXORD   IWORK(5)  THE MAXIMUM ORDER TO BE ALLOWED.  THE DEFAULT
C                    VALUE IS 12 IF METH = 1, AND 5 IF METH = 2.
C                    IF MAXORD EXCEEDS THE DEFAULT VALUE, IT WILL
C                    BE REDUCED TO THE DEFAULT VALUE.
C                    IF MAXORD IS CHANGED DURING THE PROBLEM, IT MAY
C                    CAUSE THE CURRENT ORDER TO BE REDUCED.
C
C MXSTEP   IWORK(6)  MAXIMUM NUMBER OF (INTERNALLY DEFINED) STEPS
C                    ALLOWED DURING ONE CALL TO THE SOLVER.
C                    THE DEFAULT VALUE IS 500.
C
C MXHNIL   IWORK(7)  MAXIMUM NUMBER OF MESSAGES PRINTED (PER PROBLEM)
C                    WARNING THAT T + H = T ON A STEP (H = STEP SIZE).
C                    THIS MUST BE POSITIVE TO RESULT IN A NON-DEFAULT
C                    VALUE.  THE DEFAULT VALUE IS 10.
C-----------------------------------------------------------------------
C OPTIONAL OUTPUTS.
C
C AS OPTIONAL ADDITIONAL OUTPUT FROM LSODES, THE VARIABLES LISTED
C BELOW ARE QUANTITIES RELATED TO THE PERFORMANCE OF LSODES
C WHICH ARE AVAILABLE TO THE USER.  THESE ARE COMMUNICATED BY WAY OF
C THE WORK ARRAYS, BUT ALSO HAVE INTERNAL MNEMONIC NAMES AS SHOWN.
C EXCEPT WHERE STATED OTHERWISE, ALL OF THESE OUTPUTS ARE DEFINED
C ON ANY SUCCESSFUL RETURN FROM LSODES, AND ON ANY RETURN WITH
C ISTATE = -1, -2, -4, -5, OR -6.  ON AN ILLEGAL INPUT RETURN
C (ISTATE = -3), THEY WILL BE UNCHANGED FROM THEIR EXISTING VALUES
C (IF ANY), EXCEPT POSSIBLY FOR TOLSF, LENRW, AND LENIW.
C ON ANY ERROR RETURN, OUTPUTS RELEVANT TO THE ERROR WILL BE DEFINED,
C AS NOTED BELOW.
C
C NAME     LOCATION      MEANING
C
C HU       RWORK(11) THE STEP SIZE IN T LAST USED (SUCCESSFULLY).
C
C HCUR     RWORK(12) THE STEP SIZE TO BE ATTEMPTED ON THE NEXT STEP.
C
C TCUR     RWORK(13) THE CURRENT VALUE OF THE INDEPENDENT VARIABLE
C                    WHICH THE SOLVER HAS ACTUALLY REACHED, I.E. THE
C                    CURRENT INTERNAL MESH POINT IN T.  ON OUTPUT, TCUR
```

```
C                     WILL ALWAYS BE AT LEAST AS FAR AS THE ARGUMENT
C                     T, BUT MAY BE FARTHER (IF INTERPOLATION WAS DONE).
C
C TOLSF    RWORK(14)  A TOLERANCE SCALE FACTOR, GREATER THAN 1.0,
C                     COMPUTED WHEN A REQUEST FOR TOO MUCH ACCURACY WAS
C                     DETECTED (ISTATE = -3 IF DETECTED AT THE START OF
C                     THE PROBLEM, ISTATE = -2 OTHERWISE).  IF ITOL IS
C                     LEFT UNALTERED BUT RTOL AND ATOL ARE UNIFORMLY
C                     SCALED UP BY A FACTOR OF TOLSF FOR THE NEXT CALL,
C                     THEN THE SOLVER IS DEEMED LIKELY TO SUCCEED.
C                     (THE USER MAY ALSO IGNORE TOLSF AND ALTER THE
C                     TOLERANCE PARAMETERS IN ANY OTHER WAY APPROPRIATE.)
C
C NST      IWORK(11)  THE NUMBER OF STEPS TAKEN FOR THE PROBLEM SO FAR.
C
C NFE      IWORK(12)  THE NUMBER OF F EVALUATIONS FOR THE PROBLEM SO FAR,
C                     EXCLUDING THOSE FOR STRUCTURE DETERMINATION
C                     (MOSS = 2).
C
C NJE      IWORK(13)  THE NUMBER OF JACOBIAN EVALUATIONS FOR THE PROBLEM
C                     SO FAR, EXCLUDING THOSE FOR STRUCTURE DETERMINATION
C                     (MOSS = 1).
C
C NQU      IWORK(14)  THE METHOD ORDER LAST USED (SUCCESSFULLY).
C
C NQCUR    IWORK(15)  THE ORDER TO BE ATTEMPTED ON THE NEXT STEP.
C
C IMXER    IWORK(16)  THE INDEX OF THE COMPONENT OF LARGEST MAGNITUDE IN
C                     THE WEIGHTED LOCAL ERROR VECTOR ( E(I)/EWT(I) ),
C                     ON AN ERROR RETURN WITH ISTATE = -4 OR -5.
C
C LENRW    IWORK(17)  THE LENGTH OF RWORK ACTUALLY REQUIRED.
C                     THIS IS DEFINED ON NORMAL RETURNS AND ON AN ILLEGAL
C                     INPUT RETURN FOR INSUFFICIENT STORAGE.
C
C LENIW    IWORK(18)  THE LENGTH OF IWORK ACTUALLY REQUIRED.
C                     THIS IS DEFINED ON NORMAL RETURNS AND ON AN ILLEGAL
C                     INPUT RETURN FOR INSUFFICIENT STORAGE.
C
C NNZ      IWORK(19)  THE NUMBER OF NONZERO ELEMENTS IN THE JACOBIAN
C                     MATRIX, INCLUDING THE DIAGONAL (MITER = 1 OR 2).
C                     (THIS MAY DIFFER FROM THAT GIVEN BY IA(NEQ+1)-1
C                     IF MOSS = 0, BECAUSE OF ADDED DIAGONAL ENTRIES.)
C
C NGP      IWORK(20)  THE NUMBER OF GROUPS OF COLUMN INDICES, USED IN
C                     DIFFERENCE QUOTIENT JACOBIAN APROXIMATIONS IF
C                     MITER = 2.  THIS IS ALSO THE NUMBER OF EXTRA F
C                     EVALUATIONS NEEDED FOR EACH JACOBIAN EVALUATION.
C
C NLU      IWORK(21)  THE NUMBER OF SPARSE LU DECOMPOSITIONS FOR THE
C                     PROBLEM SO FAR.
C
C LYH      IWORK(22)  THE BASE ADDRESS IN RWORK OF THE HISTORY ARRAY YH,
C                     DESCRIBED BELOW IN THIS LIST.
C
C IPIAN    IWORK(23)  THE BASE ADDRESS OF THE STRUCTURE DESCRIPTOR ARRAY
C                     IAN, DESCRIBED BELOW IN THIS LIST.
C
C IPJAN    IWORK(24)  THE BASE ADDRESS OF THE STRUCTURE DESCRIPTOR ARRAY
C                     JAN, DESCRIBED BELOW IN THIS LIST.
```

```
C
C NZL     .IWORK(25) THE NUMBER OF NONZERO ELEMENTS IN THE STRICT LOWER
C                    TRIANGLE OF THE LU FACTORIZATION USED IN THE CHORD
C                    ITERATION (MITER = 1 OR 2).
C
C NZU      IWORK(26) THE NUMBER OF NONZERO ELEMENTS IN THE STRICT UPPER
C                    TRIANGLE OF THE LU FACTORIZATION USED IN THE CHORD
C                    ITERATION (MITER = 1 OR 2).
C                    THE TOTAL NUMBER OF NONZEROS IN THE FACTORIZATION
C                    IS THEREFORE NZL + NZU + NEQ.
C
C THE FOLLOWING FOUR ARRAYS ARE SEGMENTS OF THE RWORK ARRAY WHICH
C MAY ALSO BE OF INTEREST TO THE USER AS OPTIONAL OUTPUTS.
C FOR EACH ARRAY, THE TABLE BELOW GIVES ITS INTERNAL NAME,
C ITS BASE ADDRESS, AND ITS DESCRIPTION.
C FOR YH AND ACOR, THE BASE ADDRESSES ARE IN RWORK (A REAL ARRAY).
C THE INTEGER ARRAYS IAN AND JAN ARE TO BE OBTAINED BY DECLARING AN
C INTEGER ARRAY IWK AND IDENTIFYING IWK(1) WITH RWORK(21), USING EITHER
C AN EQUIVALENCE STATEMENT OR A SUBROUTINE CALL.  THEN THE BASE
C ADDRESSES IPIAN (OF IAN) AND IPJAN (OF JAN) IN IWK ARE TO BE OBTAINED
C AS OPTIONAL OUTPUTS IWORK(23) AND IWORK(24), RESPECTIVELY.
C THUS IAN(1) IS IWK(IPIAN), ETC.
C
C NAME     BASE ADDRESS      DESCRIPTION
C
C IAN     IPIAN (IN IWK)   STRUCTURE DESCRIPTOR ARRAY OF SIZE NEQ + 1.
C JAN     IPJAN (IN IWK)   STRUCTURE DESCRIPTOR ARRAY OF SIZE NNZ.
C          (SEE ABOVE)     IAN AND JAN TOGETHER DESCRIBE THE SPARSITY
C                          STRUCTURE OF THE JACOBIAN MATRIX, AS USED BY
C                          LSODES WHEN MITER = 1 OR 2.
C                          JAN CONTAINS THE ROW INDICES OF THE NONZERO
C                          LOCATIONS, READING IN COLUMNWISE ORDER, AND
C                          IAN CONTAINS THE STARTING LOCATIONS IN JAN OF
C                          THE DESCRIPTIONS OF COLUMNS 1,...,NEQ, IN
C                          THAT ORDER, WITH IAN(1) = 1.  THUS FOR EACH
C                          J = 1,...,NEQ, THE ROW INDICES I OF THE
C                          NONZERO LOCATIONS IN COLUMN J ARE
C                          I = JAN(K),  IAN(J) .LE. K .LT. IAN(J+1).
C                          NOTE THAT IAN(NEQ+1) = NNZ + 1.
C                          (IF MOSS = 0, IAN/JAN MAY DIFFER FROM THE
C                          INPUT IA/JA BECAUSE OF A DIFFERENT ORDERING
C                          IN EACH COLUMN, AND ADDED DIAGONAL ENTRIES.)
C
C YH       LYH             THE NORDSIECK HISTORY ARRAY, OF SIZE NYH BY
C           (OPTIONAL      (NQCUR + 1), WHERE NYH IS THE INITIAL VALUE
C           OUTPUT)        OF NEQ.  FOR J = 0,1,...,NQCUR, COLUMN J+1
C                          OF YH CONTAINS HCUR**J/FACTORIAL(J) TIMES
C                          THE J-TH DERIVATIVE OF THE INTERPOLATING
C                          POLYNOMIAL CURRENTLY REPRESENTING THE SOLUTION,
C                          EVALUATED AT T = TCUR.  THE BASE ADDRESS LYH
C                          IS ANOTHER OPTIONAL OUTPUT, LISTED ABOVE.
C
C ACOR     LENRW-NEQ+1     ARRAY OF SIZE NEQ USED FOR THE ACCUMULATED
C                          CORRECTIONS ON EACH STEP, SCALED ON OUTPUT
C                          TO REPRESENT THE ESTIMATED LOCAL ERROR IN Y
C                          ON THE LAST STEP.  THIS IS THE VECTOR E IN
C                          THE DESCRIPTION OF THE ERROR CONTROL.  IT IS
C                          DEFINED ONLY ON A SUCCESSFUL RETURN FROM
C                          LSODES.
C
```

```
C-----------------------------------------------------------------------
C PART II.  OTHER ROUTINES CALLABLE.
C
C THE FOLLOWING ARE OPTIONAL CALLS WHICH THE USER MAY MAKE TO
C GAIN ADDITIONAL CAPABILITIES IN CONJUNCTION WITH LSODES.
C (THE ROUTINES XSETUN AND XSETF ARE DESIGNED TO CONFORM TO THE
C SLATEC ERROR HANDLING PACKAGE.)
C
C     FORM OF CALL                  FUNCTION
C   CALL XSETUN(LUN)                SET THE LOGICAL UNIT NUMBER, LUN, FOR
C                                   OUTPUT OF MESSAGES FROM LSODES, IF
C                                   THE DEFAULT IS NOT DESIRED.
C                                   THE DEFAULT VALUE OF LUN IS 6.
C
C   CALL XSETF(MFLAG)               SET A FLAG TO CONTROL THE PRINTING OF
C                                   MESSAGES BY LSODES.
C                                   MFLAG = 0 MEANS DO NOT PRINT. (DANGER..
C                                   THIS RISKS LOSING VALUABLE INFORMATION.)
C                                   MFLAG = 1 MEANS PRINT (THE DEFAULT).
C
C                                   EITHER OF THE ABOVE CALLS MAY BE MADE AT
C                                   ANY TIME AND WILL TAKE EFFECT IMMEDIATELY.
C
C   CALL SVCMS (RSAV, ISAV)         STORE IN RSAV AND ISAV THE CONTENTS
C                                   OF THE INTERNAL COMMON BLOCKS USED BY
C                                   LSODES (SEE PART III BELOW).
C                                   RSAV MUST BE A REAL ARRAY OF LENGTH 225
C                                   OR MORE, AND ISAV MUST BE AN INTEGER
C                                   ARRAY OF LENGTH 75 OR MORE.
C
C   CALL RSCMS (RSAV, ISAV)         RESTORE, FROM RSAV AND ISAV, THE CONTENTS
C                                   OF THE INTERNAL COMMON BLOCKS USED BY
C                                   LSODES.  PRESUMES A PRIOR CALL TO SVCMS
C                                   WITH THE SAME ARGUMENTS.
C
C                                   SVCMS AND RSCMS ARE USEFUL IF
C                                   INTERRUPTING A RUN AND RESTARTING
C                                   LATER, OR ALTERNATING BETWEEN TWO OR
C                                   MORE PROBLEMS SOLVED WITH LSODES.
C
C   CALL INTDY(,,,,,)               PROVIDE DERIVATIVES OF Y, OF VARIOUS
C        (SEE BELOW)                ORDERS, AT A SPECIFIED POINT T, IF
C                                   DESIRED.  IT MAY BE CALLED ONLY AFTER
C                                   A SUCCESSFUL RETURN FROM LSODES.
C
C THE DETAILED INSTRUCTIONS FOR USING INTDY ARE AS FOLLOWS.
C THE FORM OF THE CALL IS..
C
C   LYH = IWORK(22)
C   CALL INTDY (T, K, RWORK(LYH), NYH, DKY, IFLAG)
C
C THE INPUT PARAMETERS ARE..
C
C T         = VALUE OF INDEPENDENT VARIABLE WHERE ANSWERS ARE DESIRED
C             (NORMALLY THE SAME AS THE T LAST RETURNED BY LSODES).
C             FOR VALID RESULTS, T MUST LIE BETWEEN TCUR - HU AND TCUR.
C             (SEE OPTIONAL OUTPUTS FOR TCUR AND HU.)
C K         = INTEGER ORDER OF THE DERIVATIVE DESIRED.  K MUST SATISFY
C             0 .LE. K .LE. NQCUR, WHERE NQCUR IS THE CURRENT ORDER
C             (SEE OPTIONAL OUTPUTS).  THE CAPABILITY CORRESPONDING
```

```
C               TO K = 0, I.E. COMPUTING Y(T), IS ALREADY PROVIDED
C               BY LSODES DIRECTLY.  SINCE NQCUR .GE. 1, THE FIRST
C               DERIVATIVE DY/DT IS ALWAYS AVAILABLE WITH INTDY.
C LYH         = THE BASE ADDRESS OF THE HISTORY ARRAY YH, OBTAINED
C               AS AN OPTIONAL OUTPUT AS SHOWN ABOVE.
C NYH         = COLUMN LENGTH OF YH, EQUAL TO THE INITIAL VALUE OF NEQ.
C
C THE OUTPUT PARAMETERS ARE..
C
C DKY         = A REAL ARRAY OF LENGTH NEQ CONTAINING THE COMPUTED VALUE
C               OF THE K-TH DERIVATIVE OF Y(T).
C IFLAG       = INTEGER FLAG, RETURNED AS 0 IF K AND T WERE LEGAL,
C               -1 IF K WAS ILLEGAL, AND -2 IF T WAS ILLEGAL.
C               ON AN ERROR RETURN, A MESSAGE IS ALSO WRITTEN.
C-----------------------------------------------------------------------
C PART III.  COMMON BLOCKS.
C
C IF LSODES IS TO BE USED IN AN OVERLAY SITUATION, THE USER
C MUST DECLARE, IN THE PRIMARY OVERLAY, THE VARIABLES IN..
C   (1) THE CALL SEQUENCE TO LSODES,
C   (2) THE THREE INTERNAL COMMON BLOCKS
C         /LS0001/  OF LENGTH  258  (219 SINGLE PRECISION WORDS
C                         FOLLOWED BY 39 INTEGER WORDS),
C         /LSS001/  OF LENGTH  40    ( 6 SINGLE PRECISION WORDS
C                         FOLLOWED BY 34 INTEGER WORDS),
C         /EH0001/  OF LENGTH  2 (INTEGER WORDS).
C
C IF LSODES IS USED ON A SYSTEM IN WHICH THE CONTENTS OF INTERNAL
C COMMON BLOCKS ARE NOT PRESERVED BETWEEN CALLS, THE USER SHOULD
C DECLARE THE ABOVE THREE COMMON BLOCKS IN HIS MAIN PROGRAM TO INSURE
C THAT THEIR CONTENTS ARE PRESERVED.
C
C IF THE SOLUTION OF A GIVEN PROBLEM BY LSODES IS TO BE INTERRUPTED
C AND THEN LATER CONTINUED, SUCH AS WHEN RESTARTING AN INTERRUPTED RUN
C OR ALTERNATING BETWEEN TWO OR MORE PROBLEMS, THE USER SHOULD SAVE,
C FOLLOWING THE RETURN FROM THE LAST LSODES CALL PRIOR TO THE
C INTERRUPTION, THE CONTENTS OF THE CALL SEQUENCE VARIABLES AND THE
C INTERNAL COMMON BLOCKS, AND LATER RESTORE THESE VALUES BEFORE THE
C NEXT LSODES CALL FOR THAT PROBLEM.  TO SAVE AND RESTORE THE COMMON
C BLOCKS, USE SUBROUTINES SVCMS AND RSCMS (SEE PART II ABOVE).
C
C NOTE.. IN THIS VERSION OF LSODES, THERE ARE TWO DATA STATEMENTS,
C IN SUBROUTINES LSODES AND XERRWV, WHICH LOAD VARIABLES INTO THESE
C LABELED COMMON BLOCKS.  ON SOME SYSTEMS, IT MAY BE NECESSARY TO
C MOVE THESE TO A SEPARATE BLOCK DATA SUBPROGRAM.
C
C-----------------------------------------------------------------------
C PART IV.  OPTIONALLY REPLACEABLE SOLVER ROUTINES.
C
C BELOW ARE DESCRIPTIONS OF TWO ROUTINES IN THE LSODES PACKAGE WHICH
C RELATE TO THE MEASUREMENT OF ERRORS.  EITHER ROUTINE CAN BE
C REPLACED BY A USER-SUPPLIED VERSION, IF DESIRED.  HOWEVER, SINCE SUCH
C A REPLACEMENT MAY HAVE A MAJOR IMPACT ON PERFORMANCE, IT SHOULD BE
C DONE ONLY WHEN ABSOLUTELY NECESSARY, AND ONLY WITH GREAT CAUTION.
C (NOTE.. THE MEANS BY WHICH THE PACKAGE VERSION OF A ROUTINE IS
C SUPERSEDED BY THE USER-S VERSION MAY BE SYSTEM-DEPENDENT.)
C
C (A) EWSET.
C THE FOLLOWING SUBROUTINE IS CALLED JUST BEFORE EACH INTERNAL
C INTEGRATION STEP, AND SETS THE ARRAY OF ERROR WEIGHTS, EWT, AS
```

```
C DESCRIBED UNDER ITOL/RTOL/ATOL ABOVE..
C     SUBROUTINE EWSET (NEQ, ITOL, RTOL, ATOL, YCUR, EWT)
C WHERE NEQ, ITOL, RTOL, AND ATOL ARE AS IN THE LSODES CALL SEQUENCE,
C YCUR CONTAINS THE CURRENT DEPENDENT VARIABLE VECTOR, AND
C EWT IS THE ARRAY OF WEIGHTS SET BY EWSET.
C
C IF THE USER SUPPLIES THIS SUBROUTINE, IT MUST RETURN IN EWT(I)
C (I = 1,...,NEQ) A POSITIVE QUANTITY SUITABLE FOR COMPARING ERRORS
C IN Y(I) TO.  THE EWT ARRAY RETURNED BY EWSET IS PASSED TO THE
C VNORM ROUTINE (SEE BELOW), AND ALSO USED BY LSODES IN THE COMPUTATION
C OF THE OPTIONAL OUTPUT IMXER, THE DIAGONAL JACOBIAN APPROXIMATION,
C AND THE INCREMENTS FOR DIFFERENCE QUOTIENT JACOBIANS.
C
C IN THE USER-SUPPLIED VERSION OF EWSET, IT MAY BE DESIRABLE TO USE
C THE CURRENT VALUES OF DERIVATIVES OF Y.  DERIVATIVES UP TO ORDER NQ
C ARE AVAILABLE FROM THE HISTORY ARRAY YH, DESCRIBED ABOVE UNDER
C OPTIONAL OUTPUTS.  IN EWSET, YH IS IDENTICAL TO THE YCUR ARRAY,
C EXTENDED TO NQ + 1 COLUMNS WITH A COLUMN LENGTH OF NYH AND SCALE
C FACTORS OF H**J/FACTORIAL(J).  ON THE FIRST CALL FOR THE PROBLEM,
C GIVEN BY NST = 0, NQ IS 1 AND H IS TEMPORARILY SET TO 1.0.
C THE QUANTITIES NQ, NYH, H, AND NST CAN BE OBTAINED BY INCLUDING
C IN EWSET THE STATEMENTS..
C     COMMON /LS0001/ RLS(219),ILS(39)
C     NQ = ILS(35)
C     NYH = ILS(14)
C     NST = ILS(36)
C     H = RLS(213)
C THUS, FOR EXAMPLE, THE CURRENT VALUE OF DY/DT CAN BE OBTAINED AS
C YCUR(NYH+I)/H  (I=1,...,NEQ)  (AND THE DIVISION BY H IS
C UNNECESSARY WHEN NST = 0).

C (B) VNORM.
C THE FOLLOWING IS A REAL FUNCTION ROUTINE WHICH COMPUTES THE WEIGHTED
C ROOT-MEAN-SQUARE NORM OF A VECTOR V..
C     D = VNORM (N, V, W)
C WHERE..
C   N = THE LENGTH OF THE VECTOR,
C   V = REAL ARRAY OF LENGTH N CONTAINING THE VECTOR,
C   W = REAL ARRAY OF LENGTH N CONTAINING WEIGHTS,
C   D = SQRT( (1/N) * SUM(V(I)*W(I))**2 ).
C VNORM IS CALLED WITH N = NEQ AND WITH W(I) = 1.0/EWT(I), WHERE
C EWT IS AS SET BY SUBROUTINE EWSET.
C
C IF THE USER SUPPLIES THIS FUNCTION, IT SHOULD RETURN A NON-NEGATIVE
C VALUE OF VNORM SUITABLE FOR USE IN THE ERROR CONTROL IN LSODES.
C NONE OF THE ARGUMENTS SHOULD BE ALTERED BY VNORM.
C FOR EXAMPLE, A USER-SUPPLIED VNORM ROUTINE MIGHT..
C   -SUBSTITUTE A MAX-NORM OF (V(I)*W(I)) FOR THE RMS-NORM, OR
C   -IGNORE SOME COMPONENTS OF V IN THE NORM, WITH THE EFFECT OF
C    SUPPRESSING THE ERROR CONTROL ON THOSE COMPONENTS OF Y.
C-----------------------------------------------------------------------
C-----------------------------------------------------------------------
C OTHER ROUTINES IN THE LSODES PACKAGE.
C
C IN ADDITION TO SUBROUTINE LSODES, THE LSODES PACKAGE INCLUDES THE
C FOLLOWING SUBROUTINES AND FUNCTION ROUTINES..
C  IPREP    ACTS AS AN ITERFACE BETWEEN LSODES AND PREP, AND ALSO DOES
C           ADJUSTING OF WORK SPACE POINTERS AND WORK ARRAYS.
C  PREP     IS CALLED BY IPREP TO COMPUTE SPARSITY AND DO SPARSE MATRIX
C           PREPROCESSING IF MITER = 1 OR 2.
```

```
C  JGROUP    IS CALLED BY PREP TO COMPUTE GROUPS OF JACOBIAN COLUMN
C           INDICES FOR USE WHEN MITER = 2.
C  ADJLR     ADJUSTS THE LENGTH OF REQUIRED SPARSE MATRIX WORK SPACE.
C           IT IS CALLED BY PREP.
C  CNTNZU    IS CALLED BY PREP AND COUNTS THE NONZERO ELEMENTS IN THE
C           STRICT UPPER TRIANGLE OF J + J-TRANSPOSE, WHERE J = DF/DY.
C  INTDY     COMPUTES AN INTERPOLATED VALUE OF THE Y VECTOR AT T = TOUT.
C  STODE     IS THE CORE INTEGRATOR, WHICH DOES ONE STEP OF THE
C           INTEGRATION AND THE ASSOCIATED ERROR CONTROL.
C  CFODE     SETS ALL METHOD COEFFICIENTS AND TEST CONSTANTS.
C  PRJS      COMPUTES AND PREPROCESSES THE JACOBIAN MATRIX J = DF/DY
C           AND THE NEWTON ITERATION MATRIX P = I - H*L0*J.
C  SLSS      MANAGES SOLUTION OF LINEAR SYSTEM IN CHORD ITERATION.
C  EWSET     SETS THE ERROR WEIGHT VECTOR EWT BEFORE EACH STEP.
C  VNORM     COMPUTES THE WEIGHTED R.M.S. NORM OF A VECTOR.
C  SVCMS AND RSCMS   ARE USER-CALLABLE ROUTINES TO SAVE AND RESTORE,
C           RESPECTIVELY, THE CONTENTS OF THE INTERNAL COMMON BLOCKS.
C  ODRV      CONSTRUCTS A REORDERING OF THE ROWS AND COLUMNS OF
C           A MATRIX BY THE MINIMUM DEGREE ALGORITHM.  ODRV IS A
C           DRIVER ROUTINE WHICH CALLS SUBROUTINES MD, MDI, MDM,
C           MDP, MDU, AND SRO.  SEE REF. 2 FOR DETAILS.  (THE ODRV
C           MODULE HAS BEEN MODIFIED SINCE REF. 2, HOWEVER.)
C  CDRV      PERFORMS REORDERING, SYMBOLIC FACTORIZATION, NUMERICAL
C           FACTORIZATION, OR LINEAR SYSTEM SOLUTION OPERATIONS,
C           DEPENDING ON A PATH ARGUMENT IPATH.  CDRV IS A
C           DRIVER ROUTINE WHICH CALLS SUBROUTINES NROC, NSFC,
C           NNFC, NNSC, AND NNTC.  SEE REF. 3 FOR DETAILS.
C           LSODES USES CDRV TO SOLVE LINEAR SYSTEMS IN WHICH THE
C           COEFFICIENT MATRIX IS  P = I - CON*J, WHERE I IS THE
C           IDENTITY, CON IS A SCALAR, AND J IS AN APPROXIMATION TO
C           THE JACOBIAN DF/DY.  BECAUSE CDRV DEALS WITH ROWWISE
C           SPARSITY DESCRIPTIONS, CDRV WORKS WITH P-TRANSPOSE, NOT P.
C  R1MACH    COMPUTES THE UNIT ROUNDOFF IN A MACHINE-INDEPENDENT MANNER.
C  XERRWV, XSETUN, AND XSETF   HANDLE THE PRINTING OF ALL ERROR
C           MESSAGES AND WARNINGS.  XERRWV IS MACHINE-DEPENDENT.
C NOTE..  VNORM AND R1MACH ARE FUNCTION ROUTINES.
C ALL THE OTHERS ARE SUBROUTINES.
C
C THE INTRINSIC AND EXTERNAL ROUTINES USED BY LSODES ARE..
C ABS, AMAX1, AMIN1, FLOAT, MAX0, MIN0, MOD, SIGN, SQRT, AND WRITE.
C
C----------------------------------------------------------------------
C THE FOLLOWING CARD IS FOR OPTIMIZED COMPILATION ON LLL COMPILERS.
CLLL. OPTIMIZE
C----------------------------------------------------------------------
      EXTERNAL PRJS, SLSS
      INTEGER ILLIN, INIT, LYH, LEWT, LACOR, LSAVF, LWM, LIWM,
     1   MXSTEP, MXHNIL, NHNIL, NTREP, NSLAST, NYH, IOWNS
      INTEGER ICF, IERPJ, IERSL, JCUR, JSTART, KFLAG, L, METH, MITER,
     1   MAXORD, MAXCOR, MSBP, MXNCF, N, NQ, NST, NFE, NJE, NQU
      INTEGER IPLOST, IESP, ISTATC, IYS, IBA, IBIAN, IBJAN, IBJGP,
     1   IPIAN, IPJAN, IPJGP, IPIGP, IPR, IPC, IPIC, IPISP, IPRSP, IPA,
     2   LENYH, LENYHM, LENWK, LREQ, LRAT, LREST, LWMIN, MOSS, MSBJ,
     3   NSLJ, NGP, NLU, NNZ, NSP, NZL, NZU
      INTEGER I, I1, I2, IFLAG, IMAX, IMUL, IMXER, IPFLAG, IPGO, IREM,
     1   J, KGO, LENRAT, LENYHT, LENIW, LENRW, LENWM, LF0, LIA, LJA,
     2   LRTEM, LWTEM, LYHD, LYHN, MF1, MORD, MXHNL0, MXSTP0, NCOLM
      REAL TRET, ROWNS,
     1   CCMAX, EL0, H, HMIN, HMXI, HU, RC, TN, UROUND
      REAL CON0, CONMIN, CCMXJ, PSMALL, RBIG, SETH
```

```
      REAL ATOLI, AYI, BIG, EWTI, H0, HMAX, HMX, RH, RTOLI,
     1   TCRIT, TDIST, TNEXT, TOL, TOLSF, TP, SIZE, SUM, W0,
     2   R1MACH, VNORM
      DIMENSION MORD(2)
      LOGICAL IHIT
C-----------------------------------------------------------------------
C THE FOLLOWING TWO INTERNAL COMMON BLOCKS CONTAIN
C (A) VARIABLES WHICH ARE LOCAL TO ANY SUBROUTINE BUT WHOSE VALUES MUST
C     BE PRESERVED BETWEEN CALLS TO THE ROUTINE (OWN VARIABLES), AND
C (B) VARIABLES WHICH ARE COMMUNICATED BETWEEN SUBROUTINES.
C THE STRUCTURE OF EACH BLOCK IS AS FOLLOWS.. ALL REAL VARIABLES ARE
C LISTED FIRST, FOLLOWED BY ALL INTEGERS. WITHIN EACH TYPE, THE
C VARIABLES ARE GROUPED WITH THOSE LOCAL TO SUBROUTINE LSODES FIRST,
C THEN THOSE LOCAL TO SUBROUTINE STODE OR SUBROUTINE PRJS
C (NO OTHER ROUTINES HAVE OWN VARIABLES), AND FINALLY THOSE USED
C FOR COMMUNICATION. THE BLOCK LS0001 IS DECLARED IN SUBROUTINES
C LSODES, IPREP, PREP, INTDY, STODE, PRJS, AND SLSS. THE BLOCK LSS001
C IS DECLARED IN SUBROUTINES LSODES, IPREP, PREP, PRJS, AND SLSS.
C GROUPS OF VARIABLES ARE REPLACED BY DUMMY ARRAYS IN THE COMMON
C DECLARATIONS IN ROUTINES WHERE THOSE VARIABLES ARE NOT USED.
C-----------------------------------------------------------------------
      COMMON /LS0001/ TRET, ROWNS(209),
     1   CCMAX, EL0, H, HMIN, HMXI, HU, RC, TN, UROUND,
     2   ILLIN, INIT, LYH, LEWT, LACOR, LSAVF, LWM, LIWM,
     3   MXSTEP, MXHNIL, NHNIL, NTREP, NSLAST, NYH, IOWNS(6),
     4   ICF, IERPJ, IERSL, JCUR, JSTART, KFLAG, L, METH, MITER,
     5   MAXORD, MAXCOR, MSBP, MXNCF, N, NQ, NST, NFE, NJE, NQU

      COMMON /LSS001/ CON0, CONMIN, CCMXJ, PSMALL, RBIG, SETH,
     1   IPLOST, IESP, ISTATC, IYS, IBA, IBIAN, IBJAN, IBJGP,
     2   IPIAN, IPJAN, IPJGP, IPIGP, IPR, IPC, IPIC, IPISP, IPRSP, IPA,
     3   LENYH, LENYHM, LENWK, LREQ, LRAT, LREST, LWMIN, MOSS, MSBJ,
     4   NSLJ, NGP, NLU, NNZ, NSP, NZL, NZU

      DATA MORD(1),MORD(2)/12,5/, MXSTP0/500/, MXHNL0/10/
      DATA ILLIN/0/, NTREP/0/
C-----------------------------------------------------------------------
C IN THE DATA STATEMENT BELOW, SET LENRAT EQUAL TO THE RATIO OF
C THE WORDLENGTH FOR A REAL NUMBER TO THAT FOR AN INTEGER. USUALLY,
C LENRAT = 1 FOR SINGLE PRECISION AND 2 FOR DOUBLE PRECISION. IF THE
C TRUE RATIO IS NOT AN INTEGER, USE THE NEXT SMALLER INTEGER (.GE. 1).
C-----------------------------------------------------------------------
      DATA LENRAT/1/
C-----------------------------------------------------------------------
C BLOCK A.
C THIS CODE BLOCK IS EXECUTED ON EVERY CALL.
C IT TESTS ISTATE AND ITASK FOR LEGALITY AND BRANCHES APPROPIATELY.
C IF ISTATE .GT. 1 BUT THE FLAG INIT SHOWS THAT INITIALIZATION HAS
C NOT YET BEEN DONE, AN ERROR RETURN OCCURS.
C IF ISTATE = 1 AND TOUT = T, JUMP TO BLOCK G AND RETURN IMMEDIATELY.
C-----------------------------------------------------------------------
      IF (ISTATE .LT. 1 .OR. ISTATE .GT. 3) GO TO 601
      IF (ITASK .LT. 1 .OR. ITASK .GT. 5) GO TO 602
      IF (ISTATE .EQ. 1) GO TO 10
      IF (INIT .EQ. 0) GO TO 603
      IF (ISTATE .EQ. 2) GO TO 200
      GO TO 20
 10   INIT = 0
      IF (TOUT .EQ. T) GO TO 430
 20   NTREP = 0
```

```
C-----------------------------------------------------------------------
C BLOCK B.
C THE NEXT CODE BLOCK IS EXECUTED FOR THE INITIAL CALL (ISTATE = 1),
C OR FOR A CONTINUATION CALL WITH PARAMETER CHANGES (ISTATE = 3).
C IT CONTAINS CHECKING OF ALL INPUTS AND VARIOUS INITIALIZATIONS.
C IF ISTATE = 1, THE FINAL SETTING OF WORK SPACE POINTERS, THE MATRIX
C PREPROCESSING, AND OTHER INITIALIZATIONS ARE DONE IN BLOCK C.
C
C FIRST CHECK LEGALITY OF THE NON-OPTIONAL INPUTS NEQ, ITOL, IOPT,
C MF, ML, AND MU.
C-----------------------------------------------------------------------
      IF (NEQ(1) .LE. 0) GO TO 604
      IF (ISTATE .EQ. 1) GO TO 25
      IF (NEQ(1) .GT. N) GO TO 605
 25   N = NEQ(1)
      IF (ITOL .LT. 1 .OR. ITOL .GT. 4) GO TO 606
      IF (IOPT .LT. 0 .OR. IOPT .GT. 1) GO TO 607
      MOSS = MF/100
      MF1 = MF - 100*MOSS
      METH = MF1/10
      MITER = MF1 - 10*METH
      IF (MOSS .LT. 0 .OR. MOSS .GT. 2) GO TO 608
      IF (METH .LT. 1 .OR. METH .GT. 2) GO TO 608
      IF (MITER .LT. 0 .OR. MITER .GT. 3) GO TO 608
      IF (MITER .EQ. 0 .OR. MITER .EQ. 3) MOSS = 0
C NEXT PROCESS AND CHECK THE OPTIONAL INPUTS. ----------------------------
      IF (IOPT .EQ. 1) GO TO 40
      MAXORD = MORD(METH)
      MXSTEP = MXSTP0
      MXHNIL = MXHNL0
      IF (ISTATE .EQ. 1) H0 = 0.0E0
      HMXI = 0.0E0
      HMIN = 0.0E0
      SETH = 0.0E0
      GO TO 60
 40   MAXORD = IWORK(5)
      IF (MAXORD .LT. 0) GO TO 611
      IF (MAXORD .EQ. 0) MAXORD = 100
      MAXORD = MIN0(MAXORD,MORD(METH))
      MXSTEP = IWORK(6)
      IF (MXSTEP .LT. 0) GO TO 612
      IF (MXSTEP .EQ. 0) MXSTEP = MXSTP0
      MXHNIL = IWORK(7)
      IF (MXHNIL .LT. 0) GO TO 613
      IF (MXHNIL .EQ. 0) MXHNIL = MXHNL0
      IF (ISTATE .NE. 1) GO TO 50
      H0 = RWORK(5)
      IF ((TOUT - T)*H0 .LT. 0.0E0) GO TO 614
 50   HMAX = RWORK(6)
      IF (HMAX .LT. 0.0E0) GO TO 615
      HMXI = 0.0E0
      IF (HMAX .GT. 0.0E0) HMXI = 1.0E0/HMAX
      HMIN = RWORK(7)
      IF (HMIN .LT. 0.0E0) GO TO 616
      SETH = RWORK(8)
      IF (SETH .LT. 0.0E0) GO TO 609
C CHECK RTOL AND ATOL FOR LEGALITY. ---------------------------------------
 60   RTOLI = RTOL(1)
      ATOLI = ATOL(1)
      DO 65 I = 1,N
```

```
      IF (ITOL .GE. 3) RTOLI = RTOL(I)
      IF (ITOL .EQ. 2 .OR. ITOL .EQ. 4) ATOLI = ATOL(I)
      IF (RTOLI .LT. 0.0E0) GO TO 619
      IF (ATOLI .LT. 0.0E0) GO TO 620
 65   CONTINUE
C-----------------------------------------------------------------------
C COMPUTE REQUIRED WORK ARRAY LENGTHS, AS FAR AS POSSIBLE, AND TEST
C THESE AGAINST LRW AND LIW.  THEN SET TENTATIVE POINTERS FOR WORK
C ARRAYS.  POINTERS TO RWORK/IWORK SEGMENTS ARE NAMED BY PREFIXING L TO
C THE NAME OF THE SEGMENT.  E.G., THE SEGMENT YH STARTS AT RWORK(LYH).
C SEGMENTS OF RWORK (IN ORDER) ARE DENOTED  WM, YH, SAVF, EWT, ACOR.
C IF MITER = 1 OR 2, THE REQUIRED LENGTH OF THE MATRIX WORK SPACE WM
C IS NOT YET KNOWN, AND SO A CRUDE MINIMUM VALUE IS USED FOR THE
C INITIAL TESTS OF LRW AND LIW, AND YH IS TEMPORARILY STORED AS FAR
C TO THE RIGHT IN RWORK AS POSSIBLE, TO LEAVE THE MAXIMUM AMOUNT
C OF SPACE FOR WM FOR MATRIX PREPROCESSING.  THUS IF MITER = 1 OR 2
C AND MOSS .NE. 2, SOME OF THE SEGMENTS OF RWORK ARE TEMPORARILY
C OMITTED, AS THEY ARE NOT NEEDED IN THE PREPROCESSING.  THESE
C OMITTED SEGMENTS ARE.. ACOR IF ISTATE = 1, EWT AND ACOR IF ISTATE = 3
C AND MOSS = 1, AND SAVF, EWT, AND ACOR IF ISTATE = 3 AND MOSS = 0.
C-----------------------------------------------------------------------
      LRAT = LENRAT
      IF (ISTATE .EQ. 1) NYH = N
      LWMIN = 0
      IF (MITER .EQ. 1) LWMIN = 4*N + 10*N/LRAT
      IF (MITER .EQ. 2) LWMIN = 4*N + 11*N/LRAT
      IF (MITER .EQ. 3) LWMIN = N + 2
      LENYH = (MAXORD+1)*NYH
      LREST = LENYH + 3*N
      LENRW = 20 + LWMIN + LREST
      IWORK(17) = LENRW
      LENIW = 30
      IF (MOSS .EQ. 0 .AND. MITER .NE. 0 .AND. MITER .NE. 3)
     1   LENIW = LENIW + N + 1
      IWORK(18) = LENIW
      IF (LENRW .GT. LRW) GO TO 617
      IF (LENIW .GT. LIW) GO TO 618
      LIA = 31
      IF (MOSS .EQ. 0 .AND. MITER .NE. 0 .AND. MITER .NE. 3)
     1   LENIW = LENIW + IWORK(LIA+N) - 1
      IWORK(18) = LENIW
      IF (LENIW .GT. LIW) GO TO 618
      LJA = LIA + N + 1
      LIA = MIN0(LIA,LIW)
      LJA = MIN0(LJA,LIW)
      LWM = 21
      IF (ISTATE .EQ. 1) NQ = 1
      NCOLM = MIN0(NQ+1,MAXORD+2)
      LENYHM = NCOLM*NYH
      LENYHT = LENYH
      IF (MITER .EQ. 1 .OR. MITER .EQ. 2) LENYHT = LENYHM
      IMUL = 2
      IF (ISTATE .EQ. 3) IMUL = MOSS
      IF (MOSS .EQ. 2) IMUL = 3
      LRTEM = LENYHT + IMUL*N
      LWTEM = LWMIN
      IF (MITER .EQ. 1 .OR. MITER .EQ. 2) LWTEM = LRW - 20 - LRTEM
      LENWK = LWTEM
      LYHN = LWM + LWTEM
      LSAVF = LYHN + LENYHT
```

```
      LEWT = LSAVF + N
      LACOR = LEWT + N
      ISTATC = ISTATE
      IF (ISTATE .EQ. 1) GO TO 100
C-----------------------------------------------------------------------
C ISTATE = 3.  MOVE YH TO ITS NEW LOCATION.
C NOTE THAT ONLY THE PART OF YH NEEDED FOR THE NEXT STEP, NAMELY
C MIN(NQ+1,MAXORD+2) COLUMNS, IS ACTUALLY MOVED.
C A TEMPORARY ERROR WEIGHT ARRAY EWT IS LOADED IF MOSS = 2.
C SPARSE MATRIX PROCESSING IS DONE IN IPREP/PREP IF MITER = 1 OR 2.
C IF MAXORD WAS REDUCED BELOW NQ, THEN THE POINTERS ARE FINALLY SET
C SO THAT SAVF IS IDENTICAL TO YH(*,MAXORD+2).
C-----------------------------------------------------------------------
      LYHD = LYH - LYHN
      IMAX = LYHN - 1 + LENYHM
C MOVE YH.  BRANCH FOR MOVE RIGHT, NO MOVE, OR MOVE LEFT. --------------
      IF (LYHD) 70,80,74
 70   DO 72 I = LYHN,IMAX
        J = IMAX + LYHN - I
 72     RWORK(J) = RWORK(J+LYHD)
      GO TO 80
 74   DO 76 I = LYHN,IMAX
 76     RWORK(I) = RWORK(I+LYHD)
 80   LYH = LYHN
      IWORK(22) = LYH
      IF (MITER .EQ. 0 .OR. MITER .EQ. 3) GO TO 92
      IF (MOSS .NE. 2) GO TO 85
C TEMPORARILY LOAD EWT IF MITER = 1 OR 2 AND MOSS = 2. -----------------
      CALL EWSET (N, ITOL, RTOL, ATOL, RWORK(LYH), RWORK(LEWT))
      DO 82 I = 1,N
        IF (RWORK(I+LEWT-1) .LE. 0.0E0) GO TO 621
 82     RWORK(I+LEWT-1) = 1.0E0/RWORK(I+LEWT-1)
 85   CONTINUE
C IPREP AND PREP DO SPARSE MATRIX PREPROCESSING IF MITER = 1 OR 2. -----
      LSAVF = MIN0(LSAVF,LRW)
      LEWT = MIN0(LEWT,LRW)
      LACOR = MIN0(LACOR,LRW)
      CALL IPREP (NEQ, Y, RWORK, IWORK(LIA), IWORK(LJA), IPFLAG, F, JAC)
      LENRW = LWM - 1 + LENWK + LREST
      IWORK(17) = LENRW
      IF (IPFLAG .NE. -1) IWORK(23) = IPIAN
      IF (IPFLAG .NE. -1) IWORK(24) = IPJAN
      IPGO = -IPFLAG + 1
      GO TO (90, 628, 629, 630, 631, 632, 633), IPGO
 90   IWORK(22) = LYH
      IF (LENRW .GT. LRW) GO TO 617
C SET FLAG TO SIGNAL PARAMETER CHANGES TO STODE. ----------------------
 92   JSTART = -1
      IF (N .EQ. NYH) GO TO 200
C NEQ WAS REDUCED.  ZERO PART OF YH TO AVOID UNDEFINED REFERENCES. -----
      I1 = LYH + L*NYH
      I2 = LYH + (MAXORD + 1)*NYH - 1
      IF (I1 .GT. I2) GO TO 200
      DO 95 I = I1,I2
 95     RWORK(I) = 0.0E0
      GO TO 200
C-----------------------------------------------------------------------
C BLOCK C.
C THE NEXT BLOCK IS FOR THE INITIAL CALL ONLY (ISTATE = 1).
C IT CONTAINS ALL REMAINING INITIALIZATIONS, THE INITIAL CALL TO F,
```

```
C THE SPARSE MATRIX PREPROCESSING (MITER = 1 OR 2), AND THE
C CALCULATION OF THE INITIAL STEP SIZE.
C THE ERROR WEIGHTS IN EWT ARE INVERTED AFTER BEING LOADED.
C-----------------------------------------------------------------------
 100  CONTINUE
      LYH = LYHN
      IWORK(22) = LYH
      TN = T
      NST = 0
      H = 1.0E0
      NNZ = 0
      NGP = 0
      NZL = 0
      NZU = 0
C LOAD THE INITIAL VALUE VECTOR IN YH. --------------------------------
      DO 105 I = 1,N
 105    RWORK(I+LYH-1) = Y(I)
C  INITIAL CALL TO F.  (LF0 POINTS TO YH(*,2).) -------------------------
      LF0 = LYH + NYH
      CALL F (NEQ, T, Y, RWORK(LF0))
      NFE = 1
C LOAD AND INVERT THE EWT ARRAY.  (H IS TEMPORARILY SET TO 1.0.) -------
      CALL EWSET (N, ITOL, RTOL, ATOL, RWORK(LYH), RWORK(LEWT))
      DO 110 I = 1,N
        IF (RWORK(I+LEWT-1) .LE. 0.0E0) GO TO 621
 110    RWORK(I+LEWT-1) = 1.0E0/RWORK(I+LEWT-1)
      IF (MITER .EQ. 0 .OR. MITER .EQ. 3) GO TO 120
C  IPREP AND PREP DO SPARSE MATRIX PREPROCESSING IF MITER = 1 OR 2. -----
      LACOR = MIN0(LACOR,LRW)
      CALL IPREP (NEQ, Y, RWORK, IWORK(LIA), IWORK(LJA), IPFLAG, F, JAC)
      LENRW = LWM - 1 + LENWK + LREST
      IWORK(17) = LENRW
      IF (IPFLAG .NE. -1) IWORK(23) = IPIAN
      IF (IPFLAG .NE. -1) IWORK(24) = IPJAN
      IPGO = -IPFLAG + 1
      GO TO (115, 628, 629, 630, 631, 632, 633), IPGO
 115  IWORK(22) = LYH
      IF (LENRW .GT. LRW) GO TO 617
C CHECK TCRIT FOR LEGALITY (ITASK = 4 OR 5). --------------------------
 120  CONTINUE
      IF (ITASK .NE. 4 .AND. ITASK .NE. 5) GO TO 125
      TCRIT = RWORK(1)
      IF ((TCRIT - TOUT)*(TOUT - T) .LT. 0.0E0) GO TO 625
      IF (H0 .NE. 0.0E0 .AND. (T + H0 - TCRIT)*H0 .GT. 0.0E0)
     1   H0 = TCRIT - T
C INITIALIZE ALL REMAINING PARAMETERS. --------------------------------
 125  UROUND = R1MACH(4)
      JSTART = 0
      IF (MITER .NE. 0) RWORK(LWM) = SQRT(UROUND)
      MSBJ = 50
      NSLJ = 0
      CCMXJ = 0.2E0
      PSMALL = 1000.0E0*UROUND
      RBIG = 0.01E0/PSMALL
      NHNIL = 0
      NJE = 0
      NLU = 0
      NSLAST = 0
      HU = 0.0E0
      NQU = 0
```

```
      CCMAX = 0.3E0
      MAXCOR = 3
      MSBP = 20
      MXNCF = 10
C-----------------------------------------------------------------------
C THE CODING BELOW COMPUTES THE STEP SIZE, H0, TO BE ATTEMPTED ON THE
C FIRST STEP, UNLESS THE USER HAS SUPPLIED A VALUE FOR THIS.
C FIRST CHECK THAT TOUT - T DIFFERS SIGNIFICANTLY FROM ZERO.
C A SCALAR TOLERANCE QUANTITY TOL IS COMPUTED, AS MAX(RTOL(I))
C IF THIS IS POSITIVE, OR MAX(ATOL(I)/ABS(Y(I))) OTHERWISE, ADJUSTED
C SO AS TO BE BETWEEN 100*UROUND AND 1.0E-3.
C THEN THE COMPUTED VALUE H0 IS GIVEN BY..
C                                          NEQ
C   H0**2 = TOL / ( W0**-2 + (1/NEQ) * SUM ( F(I)/YWT(I) )**2  )
C                                           1
C WHERE   W0      = MAX ( ABS(T), ABS(TOUT) ),
C         F(I)    = I-TH COMPONENT OF INITIAL VALUE OF F,
C         YWT(I) = EWT(I)/TOL  (A WEIGHT FOR Y(I)).
C THE SIGN OF H0 IS INFERRED FROM THE INITIAL VALUES OF TOUT AND T.
C-----------------------------------------------------------------------
      LF0 = LYH + NYH
      IF (H0 .NE. 0.0E0) GO TO 180
      TDIST = ABS(TOUT - T)
      W0 = AMAX1(ABS(T),ABS(TOUT))
      IF (TDIST .LT. 2.0E0*UROUND*W0) GO TO 622
      TOL = RTOL(1)
      IF (ITOL .LE. 2) GO TO 140
      DO 130 I = 1,N
 130    TOL = AMAX1(TOL,RTOL(I))
 140  IF (TOL .GT. 0.0E0) GO TO 160
      ATOLI = ATOL(1)
      DO 150 I = 1,N
        IF (ITOL .EQ. 2 .OR. ITOL .EQ. 4) ATOLI = ATOL(I)
        AYI = ABS(Y(I))
        IF (AYI .NE. 0.0E0) TOL = AMAX1(TOL,ATOLI/AYI)
 150    CONTINUE
 160  TOL = AMAX1(TOL,100.0E0*UROUND)
      TOL = AMIN1(TOL,0.001E0)
      SUM = VNORM (N, RWORK(LF0), RWORK(LEWT))
      SUM = 1.0E0/(TOL*W0*W0) + TOL*SUM**2
      H0 = 1.0E0/SQRT(SUM)
      H0 = AMIN1(H0,TDIST)
      H0 = SIGN(H0,TOUT-T)
C ADJUST H0 IF NECESSARY TO MEET HMAX BOUND. ---------------------------
 180  RH = ABS(H0)*HMXI
      IF (RH .GT. 1.0E0) H0 = H0/RH
C LOAD H WITH H0 AND SCALE YH(*,2) BY H0. ------------------------------
      H = H0
      DO 190 I = 1,N
 190    RWORK(I+LF0-1) = H0*RWORK(I+LF0-1)
      GO TO 270
C-----------------------------------------------------------------------
C BLOCK D.
C THE NEXT CODE BLOCK IS FOR CONTINUATION CALLS ONLY (ISTATE = 2 OR 3)
C AND IS TO CHECK STOP CONDITIONS BEFORE TAKING A STEP.
C-----------------------------------------------------------------------
 200  NSLAST = NST
      GO TO (210, 250, 220, 230, 240), ITASK
 210  IF ((TN - TOUT)*H .LT. 0.0E0) GO TO 250
      CALL INTDY (TOUT, 0, RWORK(LYH), NYH, Y, IFLAG)
```

```
      IF (IFLAG .NE. 0) GO TO 627
      T = TOUT
      GO TO 420
  220 TP = TN - HU*(1.0E0 + 100.0E0*UROUND)
      IF ((TP - TOUT)*H .GT. 0.0E0) GO TO 623
      IF ((TN - TOUT)*H .LT. 0.0E0) GO TO 250
      GO TO 400
  230 TCRIT = RWORK(1)
      IF ((TN - TCRIT)*H .GT. 0.0E0) GO TO 624
      IF ((TCRIT - TOUT)*H .LT. 0.0E0) GO TO 625
      IF ((TN - TOUT)*H .LT. 0.0E0) GO TO 245
      CALL INTDY (TOUT, 0, RWORK(LYH), NYH, Y, IFLAG)
      IF (IFLAG .NE. 0) GO TO 627
      T = TOUT
      GO TO 420
  240 TCRIT = RWORK(1)
      IF ((TN - TCRIT)*H .GT. 0.0E0) GO TO 624
  245 HMX = ABS(TN) + ABS(H)
      IHIT = ABS(TN - TCRIT) .LE. 100.0E0*UROUND*HMX
      IF (IHIT) GO TO 400
      TNEXT = TN + H*(1.0E0 + 4.0E0*UROUND)
      IF ((TNEXT - TCRIT)*H .LE. 0.0E0) GO TO 250
      H = (TCRIT - TN)*(1.0E0 - 4.0E0*UROUND)
      IF (ISTATE .EQ. 2) JSTART = -2
C-----------------------------------------------------------------------
C BLOCK E.
C THE NEXT BLOCK IS NORMALLY EXECUTED FOR ALL CALLS AND CONTAINS
C THE CALL TO THE ONE-STEP CORE INTEGRATOR STODE.
C
C THIS IS A LOOPING POINT FOR THE INTEGRATION STEPS.
C
C FIRST CHECK FOR TOO MANY STEPS BEING TAKEN, UPDATE EWT (IF NOT AT
C START OF PROBLEM), CHECK FOR TOO MUCH ACCURACY BEING REQUESTED, AND
C CHECK FOR H BELOW THE ROUNDOFF LEVEL IN T.
C-----------------------------------------------------------------------
  250 CONTINUE
      IF ((NST-NSLAST) .GE. MXSTEP) GO TO 500
      CALL EWSET (N, ITOL, RTOL, ATOL, RWORK(LYH), RWORK(LEWT))
      DO 260 I = 1,N
        IF (RWORK(I+LEWT-1) .LE. 0.0E0) GO TO 510
  260   RWORK(I+LEWT-1) = 1.0E0/RWORK(I+LEWT-1)
  270 TOLSF = UROUND*VNORM (N, RWORK(LYH), RWORK(LEWT))
      IF (TOLSF .LE. 1.0E0) GO TO 280
      TOLSF = TOLSF*2.0E0
      IF (NST .EQ. 0) GO TO 626
      GO TO 520
  280 IF ((TN + H) .NE. TN) GO TO 290
      NHNIL = NHNIL + 1
      IF (NHNIL .GT. MXHNIL) GO TO 290
      CALL XERRWV(50HLSODES-- WARNING..INTERNAL T (=R1) AND H (=R2) ARE,
     1   50, 101, 1, 0, 0, 0, 0, 0.0E0, 0.0E0)
      CALL XERRWV(
     1  60H      SUCH THAT IN THE MACHINE, T + H = T ON THE NEXT STEP  ,
     1   60, 101, 1, 0, 0, 0, 0, 0.0E0, 0.0E0)
      CALL XERRWV(50H       (H = STEP SIZE). SOLVER WILL CONTINUE ANYWAY,
     1   50, 101, 1, 0, 0, 0, 2, TN, H)
      IF (NHNIL .LT. MXHNIL) GO TO 290
      CALL XERRWV(50HLSODES-- ABOVE WARNING HAS BEEN ISSUED I1 TIMES.  ,
     1   50, 102, 1, 0, 0, 0, 0, 0.0E0, 0.0E0)
      CALL XERRWV(50H       IT WILL NOT BE ISSUED AGAIN FOR THIS PROBLEM,
```

```
     1   50, 102, 1, 1, MXHNIL, 0, 0, 0.0E0, 0.0E0)
 290  CONTINUE
C-----------------------------------------------------------------------
C    CALL STODE(NEQ,Y,YH,NYH,YH,EWT,SAVF,ACOR,WM,WM,F,JAC,PRJS,SLSS)
C-----------------------------------------------------------------------
      CALL STODE (NEQ, Y, RWORK(LYH), NYH, RWORK(LYH), RWORK(LEWT),
     1   RWORK(LSAVF), RWORK(LACOR), RWORK(LWM), RWORK(LWM),
     2   F, JAC, PRJS, SLSS)
      KGO = 1 - KFLAG
      GO TO (300, 530, 540), KGO
C-----------------------------------------------------------------------
C BLOCK F.
C THE FOLLOWING BLOCK HANDLES THE CASE OF A SUCCESSFUL RETURN FROM THE
C CORE INTEGRATOR (KFLAG = 0).  TEST FOR STOP CONDITIONS.
C-----------------------------------------------------------------------
 300  INIT = 1
      GO TO (310, 400, 330, 340, 350), ITASK
C ITASK = 1.  IF TOUT HAS BEEN REACHED, INTERPOLATE. ------------------
 310  IF ((TN - TOUT)*H .LT. 0.0E0) GO TO 250
      CALL INTDY (TOUT, 0, RWORK(LYH), NYH, Y, IFLAG)
      T = TOUT
      GO TO 420
C ITASK = 3.  JUMP TO EXIT IF TOUT WAS REACHED. ------------------------
 330  IF ((TN - TOUT)*H .GE. 0.0E0) GO TO 400
      GO TO 250
C ITASK = 4.  SEE IF TOUT OR TCRIT WAS REACHED.  ADJUST H IF NECESSARY.
 340  IF ((TN - TOUT)*H .LT. 0.0E0) GO TO 345
      CALL INTDY (TOUT, 0, RWORK(LYH), NYH, Y, IFLAG)
      T = TOUT
      GO TO 420
 345  HMX = ABS(TN) + ABS(H)
      IHIT = ABS(TN - TCRIT) .LE. 100.0E0*UROUND*HMX
      IF (IHIT) GO TO 400
      TNEXT = TN + H*(1.0E0 + 4.0E0*UROUND)
      IF ((TNEXT - TCRIT)*H .LE. 0.0E0) GO TO 250
      H = (TCRIT - TN)*(1.0E0 - 4.0E0*UROUND)
      JSTART = -2
      GO TO 250
C ITASK = 5.  SEE IF TCRIT WAS REACHED AND JUMP TO EXIT. ---------------
 350  HMX = ABS(TN) + ABS(H)
      IHIT = ABS(TN - TCRIT) .LE. 100.0E0*UROUND*HMX
C-----------------------------------------------------------------------
C BLOCK G.
C THE FOLLOWING BLOCK HANDLES ALL SUCCESSFUL RETURNS FROM LSODES.
C IF ITASK .NE. 1, Y IS LOADED FROM YH AND T IS SET ACCORDINGLY.
C ISTATE IS SET TO 2, THE ILLEGAL INPUT COUNTER IS ZEROED, AND THE
C OPTIONAL OUTPUTS ARE LOADED INTO THE WORK ARRAYS BEFORE RETURNING.
C IF ISTATE = 1 AND TOUT = T, THERE IS A RETURN WITH NO ACTION TAKEN,
C EXCEPT THAT IF THIS HAS HAPPENED REPEATEDLY, THE RUN IS TERMINATED.
C-----------------------------------------------------------------------
 400  DO 410 I = 1,N
 410    Y(I) = RWORK(I+LYH-1)
      T = TN
      IF (ITASK .NE. 4 .AND. ITASK .NE. 5) GO TO 420
      IF (IHIT) T = TCRIT
 420  ISTATE = 2
      ILLIN = 0
      RWORK(11) = HU
      RWORK(12) = H
      RWORK(13) = TN
```

```
      IWORK(11) = NST
      IWORK(12) = NFE
      IWORK(13) = NJE
      IWORK(14) = NQU
      IWORK(15) = NQ
      IWORK(19) = NNZ
      IWORK(20) = NGP
      IWORK(21) = NLU
      IWORK(25) = NZL
      IWORK(26) = NZU
      RETURN
C
430   NTREP = NTREP + 1
      IF (NTREP .LT. 5) RETURN
      CALL XERRWV(
     1  60HLSODES-- REPEATED CALLS WITH ISTATE = 1 AND TOUT = T (=R1)   ,
     1    60, 301, 1, 0, 0, 0, 1, T, 0.0E0)
      GO TO 800
C-----------------------------------------------------------------------
C BLOCK H.
C THE FOLLOWING BLOCK HANDLES ALL UNSUCCESSFUL RETURNS OTHER THAN
C THOSE FOR ILLEGAL INPUT.  FIRST THE ERROR MESSAGE ROUTINE IS CALLED.
C IF THERE WAS AN ERROR TEST OR CONVERGENCE TEST FAILURE, IMXER IS SET.
C THEN Y IS LOADED FROM YH, T IS SET TO TN, AND THE ILLEGAL INPUT
C COUNTER ILLIN IS SET TO 0.  THE OPTIONAL OUTPUTS ARE LOADED INTO
C THE WORK ARRAYS BEFORE RETURNING.
C-----------------------------------------------------------------------
C THE MAXIMUM NUMBER OF STEPS WAS TAKEN BEFORE REACHING TOUT. ----------
500   CALL XERRWV(50HLSODES-- AT CURRENT T (=R1), MXSTEP (=I1) STEPS   ,
     1    50, 201, 1, 0, 0, 0, 0, 0.0E0, 0.0E0)
      CALL XERRWV(50H      TAKEN ON THIS CALL BEFORE REACHING TOUT     ,
     1    50, 201, 1, 1, MXSTEP, 0, 1, TN, 0.0E0)
      ISTATE = -1
      GO TO 580
C EWT(I) .LE. 0.0 FOR SOME I (NOT AT START OF PROBLEM). ----------------
510   EWTI = RWORK(LEWT+I-1)
      CALL XERRWV(50HLSODES-- AT T (=R1), EWT(I1) HAS BECOME R2 .LE. 0.,
     1    50, 202, 1, 1, I, 0, 2, TN, EWTI)
      ISTATE = -6
      GO TO 580
C TOO MUCH ACCURACY REQUESTED FOR MACHINE PRECISION. -------------------
520   CALL XERRWV(50HLSODES-- AT T (=R1), TOO MUCH ACCURACY REQUESTED  ,
     1    50, 203, 1, 0, 0, 0, 0, 0.0E0, 0.0E0)
      CALL XERRWV(50H      FOR PRECISION OF MACHINE..  SEE TOLSF (=R2) ,
     1    50, 203, 1, 0, 0, 0, 2, TN, TOLSF)
      RWORK(14) = TOLSF
      ISTATE = -2
      GO TO 580
C KFLAG = -1.  ERROR TEST FAILED REPEATEDLY OR WITH ABS(H) = HMIN. -----
530   CALL XERRWV(50HLSODES-- AT T(=R1) AND STEP SIZE H(=R2), THE ERROR,
     1    50, 204, 1, 0, 0, 0, 0, 0.0E0, 0.0E0)
      CALL XERRWV(50H      TEST FAILED REPEATEDLY OR WITH ABS(H) = HMIN,
     1    50, 204, 1, 0, 0, 0, 2, TN, H)
      ISTATE = -4
      GO TO 560
C KFLAG = -2.  CONVERGENCE FAILED REPEATEDLY OR WITH ABS(H) = HMIN. ----
540   CALL XERRWV(50HLSODES-- AT T (=R1) AND STEP SIZE H (=R2), THE    ,
     1    50, 205, 1, 0, 0, 0, 0, 0.0E0, 0.0E0)
      CALL XERRWV(50H      CORRECTOR CONVERGENCE FAILED REPEATEDLY     ,
     1    50, 205, 1, 0, 0, 0, 0, 0.0E0, 0.0E0)
```

```
      CALL XERRWV(30H      OR WITH ABS(H) = HMIN   ,
     1   30, 205, 1, 0, 0, 0, 2, TN, H)
      ISTATE = -5
C COMPUTE IMXER IF RELEVANT. -------------------------------------------
 560  BIG = 0.0E0
      IMXER = 1
      DO 570 I = 1,N
        SIZE = ABS(RWORK(I+LACOR-1)*RWORK(I+LEWT-1))
        IF (BIG .GE. SIZE) GO TO 570
        BIG = SIZE
        IMXER = I
 570    CONTINUE
      IWORK(16) = IMXER
C SET Y VECTOR, T, ILLIN, AND OPTIONAL OUTPUTS. ------------------------
 580  DO 590 I = 1,N
 590    Y(I) = RWORK(I+LYH-1)
      T = TN
      ILLIN = 0
      RWORK(11) = HU
      RWORK(12) = H
      RWORK(13) = TN
      IWORK(11) = NST
      IWORK(12) = NFE
      IWORK(13) = NJE
      IWORK(14) = NQU
      IWORK(15) = NQ
      IWORK(19) = NNZ
      IWORK(20) = NGP
      IWORK(21) = NLU
      IWORK(25) = NZL
      IWORK(26) = NZU
      RETURN
C-----------------------------------------------------------------------
C BLOCK I.
C THE FOLLOWING BLOCK HANDLES ALL ERROR RETURNS DUE TO ILLEGAL INPUT
C (ISTATE = -3), AS DETECTED BEFORE CALLING THE CORE INTEGRATOR.
C FIRST THE ERROR MESSAGE ROUTINE IS CALLED.  THEN IF THERE HAVE BEEN
C 5 CONSECUTIVE SUCH RETURNS JUST BEFORE THIS CALL TO THE SOLVER,
C THE RUN IS HALTED.
C-----------------------------------------------------------------------
 601  CALL XERRWV(30HLSODES-- ISTATE (=I1) ILLEGAL ,
     1   30, 1, 1, 1, ISTATE, 0, 0, 0.0E0, 0.0E0)
      GO TO 700
 602  CALL XERRWV(30HLSODES-- ITASK (=I1) ILLEGAL  ,
     1   30, 2, 1, 1, ITASK, 0, 0, 0.0E0, 0.0E0)
      GO TO 700
 603  CALL XERRWV(50HLSODES-- ISTATE .GT. 1 BUT LSODES NOT INITIALIZED ,
     1   50, 3, 1, 0, 0, 0, 0, 0.0E0, 0.0E0)
      GO TO 700
 604  CALL XERRWV(30HLSODES-- NEQ (=I1) .LT. 1     ,
     1   30, 4, 1, 1, NEQ(1), 0, 0, 0.0E0, 0.0E0)
      GO TO 700
 605  CALL XERRWV(50HLSODES-- ISTATE = 3 AND NEQ INCREASED (I1 TO I2)  ,
     1   50, 5, 1, 2, N, NEQ(1), 0, 0.0E0, 0.0E0)
      GO TO 700
 606  CALL XERRWV(30HLSODES-- ITOL (=I1) ILLEGAL   ,
     1   30, 6, 1, 1, ITOL, 0, 0, 0.0E0, 0.0E0)
      GO TO 700
 607  CALL XERRWV(30HLSODES-- IOPT (=I1) ILLEGAL   ,
     1   30, 7, 1, 1, IOPT, 0, 0, 0.0E0, 0.0E0)
```

```
      GO TO 700
  608 CALL XERRWV(30HLSODES-- MF (=I1) ILLEGAL      ,
     1   30, 8, 1, 1, MF, 0, 0, 0.0E0, 0.0E0)
      GO TO 700
  609 CALL XERRWV(30HLSODES-- SETH (=R1) .LT. 0.0   ,
     1   30, 9, 1, 0, 0, 0, 1, SETH, 0.0E0)
      GO TO 700
  611 CALL XERRWV(30HLSODES-- MAXORD (=I1) .LT. 0   ,
     1   30, 11, 1, 1, MAXORD, 0, 0, 0.0E0, 0.0E0)
      GO TO 700
  612 CALL XERRWV(30HLSODES-- MXSTEP (=I1) .LT. 0   ,
     1   30, 12, 1, 1, MXSTEP, 0, 0, 0.0E0, 0.0E0)
      GO TO 700
  613 CALL XERRWV(30HLSODES-- MXHNIL (=I1) .LT. 0   ,
     1   30, 13, 1, 1, MXHNIL, 0, 0, 0.0E0, 0.0E0)
      GO TO 700
  614 CALL XERRWV(40HLSODES-- TOUT (=R1) BEHIND T (=R2)       ,
     1   40, 14, 1, 0, 0, 0, 2, TOUT, T)
      CALL XERRWV(50H       INTEGRATION DIRECTION IS GIVEN BY H0 (=R1)   ,
     1   50, 14, 1, 0, 0, 0, 1, H0, 0.0E0)
      GO TO 700
  615 CALL XERRWV(30HLSODES-- HMAX (=R1) .LT. 0.0   ,
     1   30, 15, 1, 0, 0, 0, 1, HMAX, 0.0E0)
      GO TO 700
  616 CALL XERRWV(30HLSODES-- HMIN (=R1) .LT. 0.0   ,
     1   30, 16, 1, 0, 0, 0, 1, HMIN, 0.0E0)
      GO TO 700
  617 CALL XERRWV(50HLSODES-- RWORK LENGTH IS INSUFFICIENT TO PROCEED. ,
     1   50, 17, 1, 0, 0, 0, 0, 0.0E0, 0.0E0)
      CALL XERRWV(
     1  60H         LENGTH NEEDED IS .GE. LENRW (=I1), EXCEEDS LRW (=I2),
     1   60, 17, 1, 2, LENRW, LRW, 0, 0.0E0, 0.0E0)
      GO TO 700
  618 CALL XERRWV(50HLSODES-- IWORK LENGTH IS INSUFFICIENT TO PROCEED. ,
     1   50, 18, 1, 0, 0, 0, 0, 0.0E0, 0.0E0)
      CALL XERRWV(
     1  60H         LENGTH NEEDED IS .GE. LENIW (=I1), EXCEEDS LIW (=I2),
     1   60, 18, 1, 2, LENIW, LIW, 0, 0.0E0, 0.0E0)
      GO TO 700
  619 CALL XERRWV(40HLSODES-- RTOL(I1) IS R1 .LT. 0.0         ,
     1   40, 19, 1, 1, I, 0, 1, RTOLI, 0.0E0)
      GO TO 700
  620 CALL XERRWV(40HLSODES-- ATOL(I1) IS R1 .LT. 0.0         ,
     1   40, 20, 1, 1, I, 0, 1, ATOLI, 0.0E0)
      GO TO 700
  621 EWTI = RWORK(LEWT+I-1)
      CALL XERRWV(40HLSODES-- EWT(I1) IS R1 .LE. 0.0          ,
     1   40, 21, 1, 1, I, 0, 1, EWTI, 0.0E0)
      GO TO 700
  622 CALL XERRWV(
     1  60HLSODES-- TOUT (=R1) TOO CLOSE TO T(=R2) TO START INTEGRATION,
     1   60, 22, 1, 0, 0, 0, 2, TOUT, T)
      GO TO 700
  623 CALL XERRWV(
     1  60HLSODES-- ITASK = I1 AND TOUT (=R1) BEHIND TCUR - HU (= R2)  ,
     1   60, 23, 1, 1, ITASK, 0, 2, TOUT, TP)
      GO TO 700
  624 CALL XERRWV(
     1  60HLSODES-- ITASK = 4 OR 5 AND TCRIT (=R1) BEHIND TCUR (=R2)    ,
     1   60, 24, 1, 0, 0, 0, 2, TCRIT, TN)
```

```
      GO TO 700
  625 CALL XERRWV(
     1  60HLSODES-- ITASK = 4 OR 5 AND TCRIT (=R1) BEHIND TOUT (=R2)    ,
     1    60, 25, 1, 0, 0, 0, 2, TCRIT, TOUT)
      GO TO 700
  626 CALL XERRWV(50HLSODES-- AT START OF PROBLEM, TOO MUCH ACCURACY    ,
     1    50, 26, 1, 0, 0, 0, 0, 0.0E0, 0.0E0)
      CALL XERRWV(
     1  60H       REQUESTED FOR PRECISION OF MACHINE..  SEE TOLSF (=R1) ,
     1    60, 26, 1, 0, 0, 0, 1, TOLSF, 0.0E0)
      RWORK(14) = TOLSF
      GO TO 700
  627 CALL XERRWV(50HLSODES-- TROUBLE FROM INTDY. ITASK = I1, TOUT = R1,
     1    50, 27, 1, 1, ITASK, 0, 1, TOUT, 0.0E0)
      GO TO 700
  628 CALL XERRWV(
     1  60HLSODES-- RWORK LENGTH INSUFFICIENT (FOR SUBROUTINE PREP).    ,
     1    60, 28, 1, 0, 0, 0, 0, 0.0E0, 0.0E0)
      CALL XERRWV(
     1  60H         LENGTH NEEDED IS .GE. LENRW (=I1), EXCEEDS LRW (=I2),
     1    60, 28, 1, 2, LENRW, LRW, 0, 0.0E0, 0.0E0)
      GO TO 700
  629 CALL XERRWV(
     1  60HLSODES-- RWORK LENGTH INSUFFICIENT (FOR SUBROUTINE JGROUP). ,
     1    60, 29, 1, 0, 0, 0, 0, 0.0E0, 0.0E0)
      CALL XERRWV(
     1  60H         LENGTH NEEDED IS .GE. LENRW (=I1), EXCEEDS LRW (=I2),
     1    60, 29, 1, 2, LENRW, LRW, 0, 0.0E0, 0.0E0)
      GO TO 700
  630 CALL XERRWV(
     1  60HLSODES-- RWORK LENGTH INSUFFICIENT (FOR SUBROUTINE ODRV).    ,
     1    60, 30, 1, 0, 0, 0, 0, 0.0E0, 0.0E0)
      CALL XERRWV(
     1  60H         LENGTH NEEDED IS .GE. LENRW (=I1), EXCEEDS LRW (=I2),
     1    60, 30, 1, 2, LENRW, LRW, 0, 0.0E0, 0.0E0)
      GO TO 700
  631 CALL XERRWV(
     1  60HLSODES-- ERROR FROM ODRV IN YALE SPARSE MATRIX PACKAGE       ,
     1    60, 31, 1, 0, 0, 0, 0, 0.0E0, 0.0E0)
      IMUL = (IYS - 1)/N
      IREM = IYS - IMUL*N
      CALL XERRWV(
     1  60H       AT T (=R1), ODRV RETURNED ERROR FLAG = I1*NEQ + I2.    ,
     1    60, 31, 1, 2, IMUL, IREM, 1, TN, 0.0E0)
      GO TO 700
  632 CALL XERRWV(
     1  60HLSODES-- RWORK LENGTH INSUFFICIENT (FOR SUBROUTINE CDRV).    ,
     1    60, 32, 1, 0, 0, 0, 0, 0.0E0, 0.0E0)
      CALL XERRWV(
     1  60H         LENGTH NEEDED IS .GE. LENRW (=I1), EXCEEDS LRW (=I2),
     1    60, 32, 1, 2, LENRW, LRW, 0, 0.0E0, 0.0E0)
      GO TO 700
  633 CALL XERRWV(
     1  60HLSODES-- ERROR FROM CDRV IN YALE SPARSE MATRIX PACKAGE       ,
     1    60, 33, 1, 0, 0, 0, 0, 0.0E0, 0.0E0)
      IMUL = (IYS - 1)/N
      IREM = IYS - IMUL*N
      CALL XERRWV(
     1  60H       AT T (=R1), CDRV RETURNED ERROR FLAG = I1*NEQ + I2.    ,
     1    60, 33, 1, 2, IMUL, IREM, 1, TN, 0.0E0)
```

```
      IF (IMUL .EQ. 2) CALL XERRWV(
     1  60H         DUPLICATE ENTRY IN SPARSITY STRUCTURE DESCRIPTORS    ,
     1   60, 33, 1, 0, 0, 0, 0, 0.0E0, 0.0E0)
      IF (IMUL .EQ. 3 .OR. IMUL .EQ. 6) CALL XERRWV(
     1  60H         INSUFFICIENT STORAGE FOR NSFC (CALLED BY CDRV)       ,
     1   60, 33, 1, 0, 0, 0, 0, 0.0E0, 0.0E0)

 700  IF (ILLIN .EQ. 5) GO TO 710
      ILLIN = ILLIN + 1
      ISTATE = -3
      RETURN
 710  CALL XERRWV(50HLSODES-- REPEATED OCCURRENCES OF ILLEGAL INPUT     ,
     1   50, 302, 1, 0, 0, 0, 0, 0.0E0, 0.0E0)

 800  CALL XERRWV(50HLSODES-- RUN ABORTED.. APPARENT INFINITE LOOP      ,
     1   50, 303, 2, 0, 0, 0, 0, 0.0E0, 0.0E0)
      RETURN
 ----------------------- END OF SUBROUTINE LSODES ----------------------
      END
```